Trabajo Especial de Licenciatura en Física

APLICACIÓN DE TÉCNICAS DE APRENDIZAJE AUTOMÁTICO A LA TRANSMISIÓN DE ESTADOS CUÁNTICOS EN CADENAS DE ESPINES

Autora: Sofía Perón Santana

Directores: Dr. Omar Osenda y Dr. Martín Domínguez

Universidad Nacional de Córdoba

Facultad de Matemática, Astronomía, Física y Computación

Diciembre 2022

Agradecimientos

El presente trabajo no hubiera sido posible sin el apoyo de un montón de personas que, de alguna u otra forma, me han acompañado en este camino. Me gustaría aprovechar este pequeño apartado para darles las gracias.

A mis papás Sandra y Ricardo por la oportunidad de estudiar esta carrera. A ellos y a toda mi familia, por su apoyo y amor incondicional durante estos años.

A mis directores Martín y Omar, por la confianza, la paciencia y la buena predisposición. Gracias por permitirme participar y por acompañarme a lo largo de este trabajo que, además de interesante, ha sido sumamente divertido.

A mis amigas y amigos de la facultad, por todo su cariño, por las risas y la compañía. Gracias por cada mate, cada hora en la salita y cada recreito de estudio. Con una mención especial a Anita, que ha escuchado tantas veces la presentación de este trabajo, que debe poder recitarlo de memoria. También, a mis amigas de Mendoza que me acompañaron a la distancia en todo momento, bancando cada alegría y cada crisis.

A la universidad pública, gratuita y de calidad. En particular, a FaMAF y a su comunidad. Gran parte de los resultados de este trabajo fueron obtenidos utilizando servidores de la facultad. Gracias a Nicolás Wolowick por coordinar y poner a nuestro alcance estos recursos de cómputo. También, me gustaría agradecer a mis profes, en especial a Yami por sus consejos y contención todos estos años.

Posiblemente me esté olvidando de mucha gente porque son incontables las personas que me han ayudado a poder estar hoy acá. Estoy convencida de que todos los logros son colectivos y que tanto el aprendizaje como el conocimiento se construyen en comunidad. Es por eso que todas y cada una de esas personas forman parte de esto y les agradezco muchísimo por el cierre de este primer capítulo.

Este trabajo está dedicado a mi abuela Tita.

Resumen

Para construir cualquier tipo de hardware, es fundamental contar con protocolos de transmisión de información entre las distintas partes del mismo. En el contexto de la computación cuántica, esta transmisión se realiza enviando un estado cuántico a lo largo de una cadena de bits cuánticos o qubits. Idealmente, el estado llega de un extremo al otro de forma rápida y sin pérdida de información. La creciente posibilidad de implementar este tipo de sistemas experimentalmente ha impulsado la búsqueda de protocolos de transmisión rápidos y de alta fidelidad. Es sencillo encontrar formas de cuantificar la fidelidad de la transmisión. Sin embargo, la función asociada a la misma depende de una gran cantidad de parámetros (que, además, crece con el largo de las cadenas). Optimizar funciones de este tipo no resulta una tarea sencilla. Con esta motivación, en el presente trabajo, se plantea utilizar dos tipos de algoritmos de aprendizaje automático o Machine Learning para encontrar parámetros que maximicen la fidelidad. En primer lugar, se utilizan algoritmos genéticos para encontrar valores de acoplamientos óptimos en cadenas representadas por Hamiltonianos de tipo XX y Heisenberg. Se analizan las diferencias de complejidad entre ambos modelos y se estudian las ventajas de agregar (o no) información física del sistema al algoritmo. En segundo lugar, se implementa un algoritmo de aprendizaje por refuerzos profundo, es decir, combinado con una red neuronal (Deep Reinforcement Learning o DRL) para encontrar secuencias de pulsos magnéticos externos que permitan mejorar la transmisión en cadenas de tipo XX.

Palabras Clave: Transmisión de Estados Cuánticos, Optimización, Aprendizaje Automático, Cadenas de Espín, Qubits, Aprendizaje por Refuerzos

Abstract

To build any kind of computer hardware, it is necessary to implement information transmission protocols between its different parts. In the context of quantum computing, this transmission consists of sending a quantum state through a Qubit chain. Ideally, the state will travel across the chain fast and without loss of information. The increasing possibilities of implementing this type of systems experimentally has inspired the search of fast, high-fidelity transfer protocols. It is easy to find a function that quantifies transmission fidelity. However, said function depends on a really large amount of parameters that increases with the length of the chain. The optimization of this type of functions is not a trivial task. With this in mind, this work explores applying two Machine Learning algorithms to find parameters to maximize fidelity. In the first place, genetic algorithms are used to find coupling values in chains described by Heisenberg and XX Hamiltonians. The complexity differences between both models are discussed and the advantages of adding to the model information regarding the physical characteristics of the system. Then, a Deep Reinforcement learning algorithm is applied to find sequences of magnetic pulses that allow transmissions in XX chains.

Keywords: Quantum State Transfer, Optimization, Machine Learning, Spin Chains, Qubits, Reinforcement Learning

Contenidos

Introducción

Así como el bit es la unidad fundamental en los sistemas de computación e información clásica, la computación e información cuánticas se construyen a partir de los bits cuánticos o "qubits" [1]. Mientras que un bit se caracteriza con un valor o estado dado por 1 o 0, un qubit se corresponde con un estado cuántico, representado matemáticamente por un vector. De manera análoga a los estados que puede tomar el bit, se define la base de estados cuánticos computacionales como aquella conformada por dos vectores que se denotan $|0\rangle$ y $|1\rangle$. Los qubits además de poder encontrarse en alguno de los dos miembros de la base, podrán adoptar cualquier combinación lineal (usualmente denominada superposición) de los mismos. Es decir, podrán tomar cualquier estado $|\psi\rangle$ de la forma:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad \alpha, \beta \in \mathbb{C} \tag{1}$$

Como sistema físico, un qubit es simplemente un sistema cuántico de dos niveles. Algunos ejemplos típicos de entidades utilizadas como qubits son los espines $1/2$ con sus dos estados posibles, o los fotones donde los estados 0 y 1 se corresponden con su polarización horizontal y vertical. De todas formas, existe una amplia variedad de sistemas que pueden utilizarse con este propósito [2].

Un sistema de múltiples qubits se denomina registro o registrador cuántico. Al igual que en cualquier tipo de hardware, la escalabilidad es una propiedad fundamental para la implementación física de la computación cuántica. Es necesario contar con la capacidad de transmitir qubits entre los distintos registradores o procesadores para poder construir procesadores más grandes. Además, se buscan protocolos que permitan que esta transmisión sea tan rápida y fiel como sea posible. Dado que las unidades utilizadas como qubits generalmente son estáticas, este transporte de información puede realizarse a través de interacciones entre los elementos de la cadena sin necesidad de que los mismos se muevan [3].

Idealmente, se busca implementar protocolos de transmisión de mínimo control, es decir, que no requieran control permanente de todos, o al menos de la mayoría, de los qubits en el sistema. Para estudiar esta motivación en detalle, se considera un sistema básico como el que se ilustra en la figura 1. El mismo está formado por dos registradores cuánticos conectados por una cadena central de espines. Los qubits en los registradores pueden ser controlados con campos magnéticos externos (representados por las flechas azules), que dan lugar a interacciones o acoplamientos variables. Sin embargo, los controles

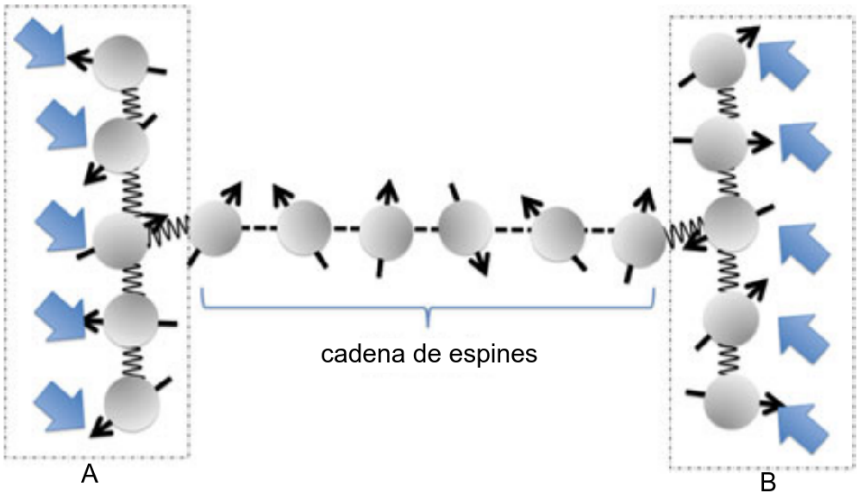

Figura 1: Registradores cuánticos (A,B) con acoplamientos (líneas onduladas) que pueden ser controlados a través de campos (flechas azules). Están conectados por una cadena horizontal de espines con interacciones permanentes (línea punteada)[4]

de los mismos deberán estar conectados a algún tipo de sistema macroscópico accesible a un humano, por ejemplo, teclas o diales. Estos, naturalmente, ocupan un espacio físico mucho mayor al de un qubit. Por eso, es fundamental poder tener regiones que transmitan información entre registradores de forma fiel sin necesidad de ese tipo de manipulación.

En la figura 1, la cadena de espines actúa como transmisor de información, lo que usualmente se conoce como *bus de datos*. El estudio del aprovechamiento de la dinámica de no equilibrio de las cadenas de espín para el manejo y transporte de información cuántica está recibiendo cada vez más atención debido a la creciente posibilidad de construir estos sistemas y configurar sus interacciones.

En este trabajo se analizan dos tipos de protocolos correspondientes a diferentes estrategias de transmisión en cadenas de espín. En primer lugar, se estudian sistemas independientes del tiempo donde las interacciones entre sus elementos se configuran de forma óptima para el transporte de información. Luego, se consideran cadenas cuyos miembros tienen interacciones fijas y se aplican campos magnéticos, externos y variables en el tiempo, sobre los espines en los extremos. Para ambos se estudia la fidelidad de las transmisiones, es decir, su capacidad de transmitir sin pérdida de información. Como se verá más adelante, es posible cuantificar esta propiedad a través de una función matemática. Así, el objeto principal de estudio serán las formas de maximizar el valor de esta función en distintos sistemas sin sacrificar la rapidez con que se completa el protocolo.

En general, la optimización en problemas de muchos cuerpos en mecánica cuántica representa un gran desafío. Esto se debe a que su complejidad crece exponencialmente con el número de parámetros. El aumento actual del poder de cómputo está impulsando la implementación de métodos de optimización cada vez más sofisticados. Sin embargo, algunas de las técnicas más utilizadas, como por ejemplo, técnicas estocásticas de tipo Montecarlo aún fallan para describir algunos sistemas [5]. Encontrar una estrategia

más general para reducir la complejidad presente en estos problemas continúa siendo un objetivo fundamental.

Las aplicaciones de los algoritmos de aprendizaje automático y redes neuronales son cada vez más variadas. En la actualidad, sus campos de acción van desde reconocimiento de texto, voz e imágenes [6] hasta la capacidad de "entrenamiento" para aprender a jugar un juego de mesa [7]. Además, están siendo utilizados en distintas áreas del conocimiento como la medicina [8], la electrónica [9] o incluso la música [10]. Las capacidades de estos métodos para el reconocimiento de patrones y la extracción de características han inspirado también su aplicación para el estudio de sistemas cuánticos de varios cuerpos, entre otras áreas de la física [5].

Se busca mostrar cómo es posible aplicar estas técnicas al problema de transmisión de estados cuánticos (*QST: Quantum State Transfer*) planteándolo como un problema de optimización sobre la fidelidad. En particular, en este trabajo, se pone el foco en dos algoritmos. En primer lugar, se estudia la implementación del algoritmo genético para encontrar acoplamientos permanentes, es decir, independientes del tiempo. Este algoritmo, de tipo poblacional, permite explorar de manera aleatoria distintos conjuntos de parámetros. Inspirado en la teoría de la evolución, mantiene a lo largo de las instancias únicamente los valores que resultan más "aptos", lo que en este caso equivale a mantener solo aquellos que maximizan la fidelidad. En segundo lugar, para el modelo de campos magnéticos, se implementa un algoritmo de aprendizaje por refuerzos. En este caso, los acoplamientos están fijos, y la optimización se realiza sobre los campos aplicados a lo largo del tiempo. Este tipo de aprendizaje automático, basado en "prueba y error", permite entrenar al sistema para que aprenda qué secuencia de pulsos magnéticos permite una mejor transmisión.

Parte I

Marco Teórico

Capítulo 1

Modelo Físico

1.1 Protocolo básico de transmisión de estados

Para ilustrar el concepto de transmisión de estados, se considera un protocolo básico de comunicación cuántica. El objetivo es describir cómo puede utilizarse la dinámica de no equilibrio de un Hamiltoniano asociado a una cadena de espines para transmitir un estado cuántico a lo largo de la misma.

Sin considerar por ahora ningún tipo de optimización de la transferencia, es posible plantear una cadena de N espines con N-1 coeficientes de intercambio J homogéneos, es decir, idénticos para todos los sitios. El Hamiltoniano correspondiente será:

$$\mathbf{H} = J \sum_{j=1}^{N-1} \boldsymbol{\sigma}_j . \boldsymbol{\sigma}_{j+1} = J \sum_{j=1}^{N-1} \left(\sigma_j^x \sigma_{j+1}^x + \sigma_j^y \sigma_{j+1}^y + \sigma_j^z \sigma_{j+1}^z \right) \tag{1.1}$$

donde $\sigma_j^{x/y/z}$ representan las matrices de Pauli actuando sobre el j-ésimo espín. Este esquema fue propuesto en el primer trabajo, realizado por Bose et al., que exploró la idea de utilizar cadenas de espines como canales de comunicación [3].

Se inicializa la cadena en algún estado simple, como por ejemplo, uno en que todos los espines estén apuntando hacia abajo, es decir, en su estado 'down'. Se elige $J < 0$ para que esta inicialización sea fácil de lograr, de modo que el sistema es ferromagnético.

En un extremo de la cadena se sitúa un estado cuántico arbitrario, dejando al resto apuntando hacia abajo. Debido a la evolución propia del sistema, este se dispersa y se propaga a lo largo de la cadena. En la figura 1.1, se ilustra cómo el estado que está en el primer espín para el estado inicial (figura superior) se propaga por el sistema hasta que para un tiempo óptimo llega al espín del extremo B (figura inferior). Al hablar de tiempo óptimo, se hace referencia al tiempo para el que el estado en el extremo B es lo más parecido al que se envió desde el extremo A. En ese momento, el receptor (que podría ser, por ejemplo, uno de los procesadores que se desea comunicar) "toma" ese estado y el protocolo de comunicación finaliza.

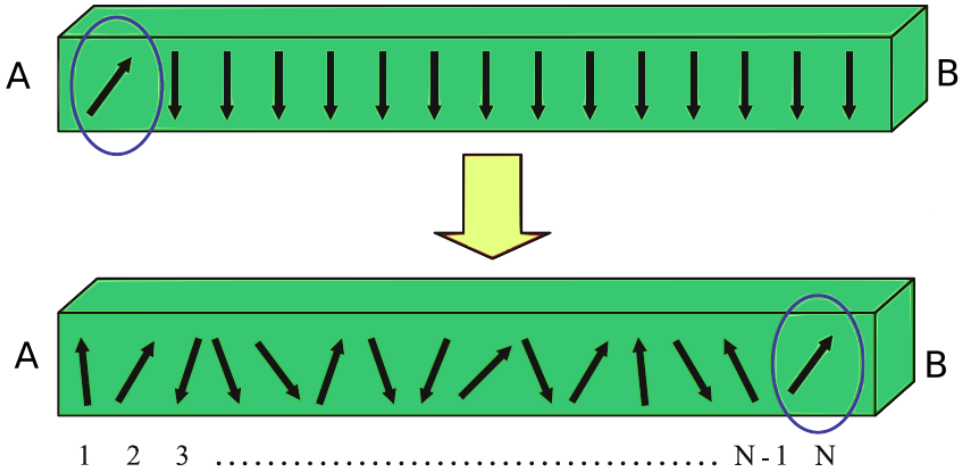

Figura 1.1: Se muestran el estado inicial y final de una cadena ferromagnética de espines 1/2 utilizada como canal básico de transmisión. El espín en el sitio 1 (extremo A) se inicializa en algún estado. Después de cierto tiempo, ese estado se transmite al espín en el sitio N (extremo B) que reproduce al menos una aproximación cercana del estado que anteriormente estaba en el primer sitio. La figura se extrajo de la Ref. [4].

Aclaración sobre unidades:

En este trabajo, se consideran sistemas cerrados, es decir, sistemas cuya dinámica es unitaria. Por lo tanto, los operadores que dictan su evolución toman la forma:

$$U(\hat{H}, t) = e^{-i\hat{H}t/\hbar} \tag{1.2}$$

Se sabe que el operador exponencial debe tener un argumento adimensional. La constante de Planck tiene unidades de momento angular, o equivalentemente, de energía multiplicada por tiempo:

$$[\hbar] = [E.\ T] = [M.\ L^2.\ T^{-1}] \tag{1.3}$$

Por otro lado, los Hamiltonianos tienen unidades de energía, al igual que los coeficientes de intercambio o acoplamientos J. Con esto en mente, a lo largo de este trabajo se utilizan unidades tales que $\hbar = 1$. Consecuentemente, las unidades resultantes de energía serán equivalentes a unidades de frecuencia permitiendo trabajar con estos operadores de forma adimensional.

Si bien este trabajo se enfoca en resultados teóricos, existen otros ([11] y sus referencias) que discuten la practicabilidad de este tipo de sistemas, en ellos los valores posibles de los coeficientes de intercambio pueden llegar hasta, aproximadamente, 40 μeV. Correspondientemente, los tiempos de transmisión resultan del orden de 10^{-9} s.

1.1.1 Fidelidad en la transmisión

Para cuantificar la calidad en la transmisión de información, es necesario definir algún tipo de figura de mérito. Cuando Bose [3] propuso la implementación de este sistema, también introdujo la fidelidad con este propósito.

Los Hamiltonianos de los sistemas estudiados en este trabajo (y, en general, la mayoría de los modelos implementados para la transmisión de estados cuánticos) conmutan con el momento angular de espín en la dirección z ($S_z = \sum_i \sigma_i^z$), es decir, cumplen con la relación de conmutación:

$$[S_z, \hat{H}] = 0 \tag{1.4}$$

Esta propiedad hace que el Hamiltoniano pueda diagonalizarse en subespacios con un número fijo de excitaciones. Dado que el interés es transmitir estados con un único espín excitado, se elige una base que se conoce como base computacional. Esta estará compuesta de los N estados con un único estado "up" en el sitio j (j=1,...,N):

$$|\boldsymbol{j}\rangle \;=\; |\downarrow\downarrow\downarrow\downarrow\;\ldots\uparrow_j\downarrow\downarrow\downarrow\downarrow\downarrow\rangle \tag{1.5}$$

Además, se define el estado $|\mathbf{0}\rangle$ como aquel con todos los espines "down". Este último, se corresponde con uno de los estados fundamentales del Hamiltoniano ferromagnético.

Con estos conceptos ya definidos, es posible estudiar con mayor detalle el protocolo de transmisión descripto anteriormente. En el tiempo $t = 0$ se sitúa en el extremo A de la cadena el estado que se desea transmitir. Este puede escribirse como:

$$|\psi_{in}\rangle = \cos(\theta/2)\,|\mathbf{0}\rangle + e^{i\phi}\sin(\theta/2)\,|\mathbf{1}\rangle \tag{1.6}$$

El operador evolución asociado al Hamiltoniano $H(\mathbf{J}^N)$ (independiente del tiempo) del sistema tiene la forma:

$$U(\mathbf{J}^N, t) = e^{-i\hat{H}(\mathbf{J}^N)t} \tag{1.7}$$

Si se consideran sistemas para los que se cumple la relación de conmutación (1.4), la componente $|\mathbf{1}\rangle$ evoluciona únicamente a estados $|\boldsymbol{j}\rangle$. Además, se tiene que la componente en $|\mathbf{0}\rangle$ se conserva. Así, el estado en un tiempo t estará dado por:

$$|\psi(t)\rangle = \cos(\theta/2)\,|\mathbf{0}\rangle + \sin(\theta/2)U(\mathbf{J}^N, t)\,|\mathbf{1}\rangle \tag{1.8}$$

Entonces, el interés radica en encontrar un propagador tal que:

$$U(\mathbf{J}^N, t)\,|\mathbf{1}\rangle = e^{-i\phi}\,|\boldsymbol{N}\rangle \tag{1.9}$$

Es decir, se busca un operador de evolución que permita transmitir la excitación del primer al último sitio (notar que la fase ϕ no es relevante para la transmisión).

Con esto en mente, se define la fidelidad como la probabilidad de transición entre esos dos estados:

$$F_N(t) = |\langle \boldsymbol{N}| \exp\{-i\boldsymbol{H}t\} |\mathbf{1}\rangle|^2 \tag{1.10}$$

En resumen, un buen protocolo de transmisión será aquel que maximice esta amplitud de probabilidad para el menor tiempo t posible. Se definen tres escenarios posibles asociados al éxito en la transmisión:

- Transmisión perfecta o PST (*Perfect State Transfer*): Existe al menos un tiempo t_{PT} tal que la fidelidad en la transmisión es igual a 1.
- Transmisión suficientemente buena, también conocida como PGST (*Pretty Good State Transfer*): Si para todo $\varepsilon > 0$ existe un tiempo t_ε tal que la fidelidad en la transmisión sea igual a $1 - \varepsilon$.
- Transmisión casi perfecta, o APST (*Almost Perfect State Transfer*): Si para algún δ fijado previamente, existe algún tiempo tal que la fidelidad en la transmisión sea mayor o igual a $1 - \delta$.

A su vez, existen dos tipos principales de protocolos para alcanzar una transmisión exitosa a través de la dinámica unitaria de una cadena de espín [12]. Por un lado, se tienen aquellos basados en el control o configuración de las interacciones entre los elementos de la cadena. En ellos, tal como se describió hasta ahora, un estado se sitúa en un extremo y se transmite al otro a través de la dinámica propia del sistema. En general, estos modelos son independientes del tiempo y su evolución está dictada únicamente por los coeficientes de intercambio o acoplamientos J_i. La segunda estrategia posible es controlar el comportamiento a través de la aplicación de campos magnéticos. En este caso, el sistema evoluciona de acuerdo a su dinámica natural sumada a forzamientos externos variables en el tiempo.

1.2 Estrategias de Transmisión y Modelos de Espín

En esta sección, se discuten distintos protocolos que implementan la primera estrategia nombrada, es decir, aquellos en los que la transmisión se realiza a través de las interacciones propias del sistema. Para definir y comparar algunos de los modelos y estrategias utilizados en trabajos previos, se considera el siguiente Hamiltoniano general para cadenas de N espines 1/2 con interacciones a primeros vecinos:

$$H = \sum_{i=1}^{N} J_i[(1+\gamma)\sigma_i^x\sigma_{i+1}^x + (1-\gamma)\sigma_i^y\sigma_{i+1}^y + \Delta\sigma_i^z\sigma_{i+1}^z] + \sum_{i=1}^{N} h_i\sigma_i^z \tag{1.11}$$

A partir de esta expresión, pueden caracterizarse una amplia variedad de sistemas que se corresponden con diferentes protocolos de comunicación [4]. En primer lugar, es posible distinguir entre los sistemas con coeficientes de intercambio h_i y J_i dependientes del sitio

i que se denominan inhomogéneos, y aquellos que no tienen dependencia del sitio que se denominan homogéneos.

Por otro lado, se tienen γ y Δ que son parámetros de anisotropía y determinan la simetría de las interacciones. Por ejemplo, para $\gamma = 0$ y $\Delta = 1$ se tiene el Hamiltoniano de Heisenberg original también conocido como modelo XXX dado que todas las componentes de espín experimentan el mismo acoplamiento a primeros vecinos, es decir, es isotrópico. Este modelo fue el estudiado en la primera propuesta [3] para la implementación de una cadena de espines como medio para el transporte de información cuántica y es el que se describe en la sección introductoria.

Para $\gamma = 0$ pero $\Delta \neq 1$ se tiene el modelo XXZ donde se destaca el caso de $\Delta = 0$ conocido como modelo XX. Se verá en secciones posteriores que configurando las interacciones individuales entre los espines, este modelo permite transmitir de forma perfecta un estado a lo largo de cadenas de cualquier tamaño.

En general, $\gamma = 0$ implica que se cumple la relación de conmutación (1.4), es decir, que se conserva la componente total de espín en z. Como se mencionó anteriormente, esto se traduce en que la cantidad de espines excitados en la cadena se mantenga constante. Es por eso que los sistemas con esta propiedad resultan de suma importancia en el estudio de la transmisión de información y son los que se analizan en este trabajo.

Otros modelos incluyen, por ejemplo, el caso con $\gamma \neq 0$ y $\Delta = 0$ que se conoce como cadena XY. En particular para $\gamma = \pm 1$ se obtiene un Hamiltoniano que representa una cadena de Ising en un campo transversal magnético. Por último, se tiene el caso del modelo general XYZ con $\gamma \neq 0$ y $\Delta \neq 0, 1$ cuyo estado fundamental se relaciona con las propiedades térmicas del modelo 2-dimensional de ocho vértices [13][14]. Más información sobre la implementación de estos modelos junto con una revisión histórica de las estrategias de transmisión utilizadas puede encontrarse en [15], [4] y sus referencias.

Se ha demostrado [16] [17] que las cadenas homogéneas no permiten una transmisión perfecta debido a efectos de dispersión. Es por eso que muchas estrategias se centran en la configuración individual de los coeficientes de intercambio o acoplamientos. A su vez, se distinguen dos opciones típicas para dicha configuración (figura 1.2). Por un lado, se tienen las cadenas completamente configuradas ('fully engineered chains') donde se ajustan los acoplamientos correspondientes a las interacciones de cada uno de los elementos de la cadena. En segundo lugar, se tiene el control o configuración de bordes, donde los acoplamientos del centro de la cadena se mantienen constantes y solo se configuran los valores de los extremos.

En la siguiente sección, se estudia la implementación de ambos protocolos para el Hamiltoniano XX. Primero, se analiza cómo puede obtenerse transmisión perfecta (PST) a través de la configuración individual de todos los acoplamientos. Luego, se describe brevemente el protocolo de configuración en los bordes que permite obtener una transmisión de alta fidelidad (PGST) modificando la intensidad de interacción de los extremos de la cadena. Por último, se analiza la implementación del esquema completamente controlado en el modelo de Heisenberg.

A lo largo de este trabajo, se consideran cadenas centro-simétricas, es decir, aquellas en que cumplen la condición:

$$J_i = J_{N-i} \tag{1.12}$$

Esta suposición garantiza que los autoestados asociados tengan paridades alternantes que es una propiedad necesaria para la transmisión perfecta [4].

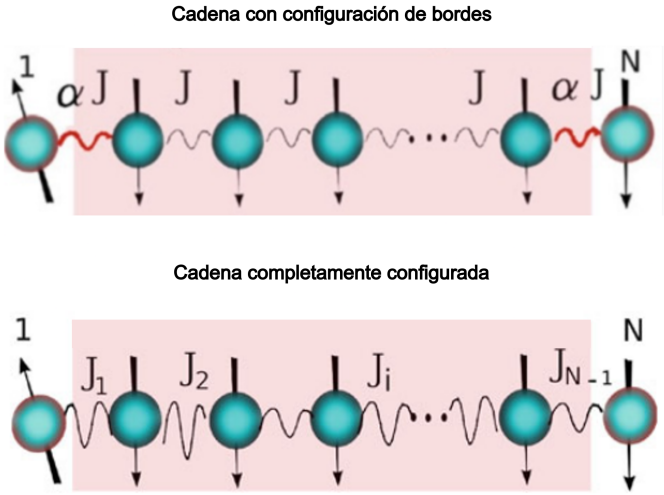

Figura 1.2: Comparación de protocolos de diseño en cadenas de N espines. En la figura superior, se muestra el esquema de control en los bordes. Los acoplamientos del centro toman un valor constante, mientras que el valor de los acoplamientos de los extremos se optimiza sobre el parámetro α. En la figura inferior, se esquematiza el esquema completamente controlado. En él, cada una de las interacciones individuales es configurada para obtener una buena transmisión.

1.3 Comparación de los protocolos de transmisión en el Hamiltoniano XX.

El Hamiltoniano XX ($\gamma = \Delta = 0$ en (1.11)) definido según:

$$H_{xx}(\mathbf{J^N}) \;=\; -\sum_{i=1}^{N-1} J_{i,i+1}\left(\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y\right) \tag{1.13}$$

es uno de los sistemas más estudiados ([12], [18], [19], [17]) en el contexto de la transmisión de estados. Dado que cumple con la relación de conmutación (ec. 1.4), puede expresarse en la base de una excitación como[1]:

[1]La deducción detallada de esta expresión puede encontrarse en la primera sección del Apéndice A.

$$H_{xx}(\mathbf{J^N}) = -\frac{1}{2}\begin{bmatrix} 0 & J_1 & & ... & 0 \\ J_1 & 0 & J_2 & ... & 0 \\ 0 & J_2 & 0 & ... & \vdots \\ 0 & \vdots & \vdots & \ddots & J_{N-1} \\ 0 & 0 & 0 & J_{N-1} & 0 \end{bmatrix} \tag{1.14}$$

1.3.1 Transmisión Perfecta a través de la configuración individual de los acoplamientos.

Para analizar cómo obtener una transmisión perfecta en este sistema, se parte del estudio de sistemas con dinámica periódica. En general, un sistema cuántico genera dinámica periódica si su espectro de energía muestra diferencias finitas. Un ejemplo típico de este comportamiento es el oscilador armónico cuyo espectro está dado por $E_n = \hbar\omega(n + 1/2)(n \geq 0)$.

Además, puede demostrarse que, bajo una fuerza armónica, una función de onda arbitraria evoluciona cumpliendo la relación:

$$\psi(x, t + \frac{\pi}{\omega}) = -i\psi(-x, t) \tag{1.15}$$

Esto implica que, en medio período, el estado del oscilador evoluciona a una imagen espejada exacta de su estado inicial. La razón de este comportamiento se relaciona con las diferencias finitas en el espectro de energías y las paridades alternadas de los auto-estados sucesivos. Surge entonces la pregunta de si llevar estas dos propiedades a un arreglo de qubits permitirá lograr una transmisión perfecta [4].

Con esto en consideración, se retoma nuevamente el estudio del modelo XX dado por (1.13), imponiendo al mismo la condición de simetría dada por (1.12). Se toma inicialmente el estado $|1\rangle$, cuya evolución está dada por:

$$|\psi(t)\rangle = U(t, \mathbf{J^N}) |1\rangle = e^{-i\hat{H}} |1\rangle \tag{1.16}$$

donde el vector $\mathbf{J^N}$ contiene los acoplamientos. A través de una descomposición espectral del Hamiltoniano en cuestión, puede expresarse el estado evolucionado como:

$$|\psi(t)\rangle = \left(\sum_k e^{-iE_k t} |k\rangle \langle k|\right) |1\rangle = |\psi(t)\rangle = \sum_k e^{-iE_k t} \langle k|1\rangle |k\rangle \tag{1.17}$$

Definiendo la proyecciones $P_{k,1} = |\langle k|1\rangle|^2$, se obtiene la siguiente expresión para la fidelidad:

$$F_N(t) = \sum_k (-1)^k P_{k,1} e^{-iE_k t} \tag{1.18}$$

Esta expresión ilustra la dependencia de la fidelidad con la estructura del espectro de energías. La transmisión perfecta ocurre si para un tiempo t_{PST} todas las contribuciones a la suma (1.18) estén en fase. Esto, solo sucede si:

$$E_{k+1} - E_k = \pi(2 * m_k + 1)/t_{PST} \tag{1.19}$$

donde el conjunto E_k de auto-energías es ordenado y m_k son enteros arbitrarios. Para una transmisión perfecta, esta condición debe cumplirse para todo par de energías sucesivas. Si esto se cumple para t_{PST}, dada la periodicidad de la condición, se producirá transmisión perfecta nuevamente en todos sus múltiplos enteros.

Una partícula cargada de espín J en un campo magnético en la dirección z tiene un espectro de $2J + 1$ niveles de energía equidistantes correspondientes con los autoestados de J_z. A su vez, las transiciones entre dichos estados se relacionan con las componentes transversales de J: J_x y J_y. Utilizando analogías de este sistema con el modelo XX, en el trabajo de Christandl et al. [17] se muestra que puede lograrse una transmisión perfecta en este tipo de cadenas. Para encontrar los acoplamientos óptimos, se imponen reglas sobre el espectro de energías y luego se resuelve un problema de autovalores inverso. De esa forma, se obtienen los acoplamientos:

$$J_i = \tilde{J}\sqrt{i(N-i)} \tag{1.20}$$

donde $\tilde{J}$ representa alguna constante de proporcionalidad. Notar que los acoplamientos resultantes cumplen con la condición (1.12).

En una primera etapa de este trabajo, correspondiente a la exploración del problema, se reprodujeron numéricamente los espectros obtenidos al diagonalizar el Hamiltoniano XX construído con los acoplamientos en (1.20). En la figura 1.3 se muestran los espectros obtenidos para cadenas de 15 y 20 espines de largo. Se corrobora la obtención de un espectro ordenado con diferencias constantes entre las auto-energías.

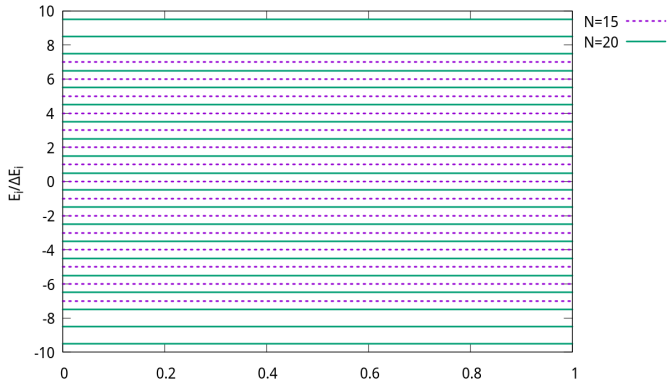

Figura 1.3: Espectros de energía obtenidos con los acoplamientos del trabajo [17] para cadena de 15 y 20 espines.

Además, se simuló la evolución temporal para este régimen. Se muestra el resultado

en la figura 1.4, donde se observa la periodicidad con la que el sistema alcanza transmisión perfecta.

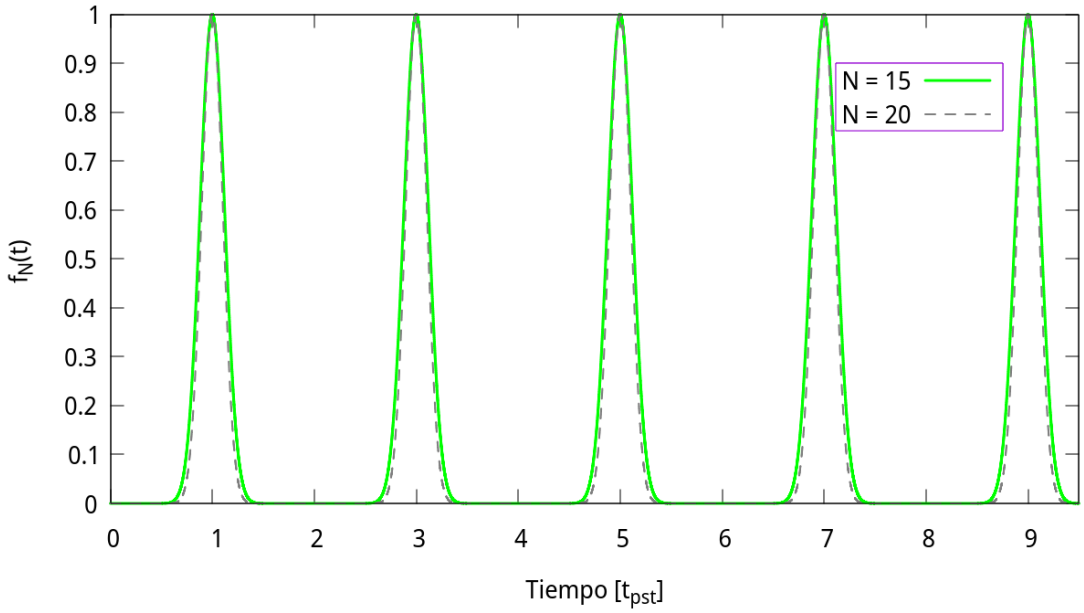

Figura 1.4: Evolución de la amplitud de probabilidad para cadena de 15 y 20 espines.

1.3.2 Alta fidelidad a través de configuración en los bordes.

En segundo lugar, se tiene la estrategia de configuración en los bordes que utiliza cadenas construidas según:

$$J_1 = J_{N-1} = \alpha J \quad ; \quad J_i = J \quad \forall i \neq 1, N-1 \tag{1.21}$$

El parámetro α modifica la fuerza de interacción de los espines en los extremos con sus vecinos más cercanos. Este tipo de sistemas permiten lograr transmisiones de alta fidelidad (APST). Dentro de este régimen, pueden mencionarse dos ejemplos. Por un lado, los sistemas de acoplamientos débiles en los que los qubits de los extremos están acoplados al centro de la cadena por valores de $\alpha < 1$. En segundo lugar, se encuentran los acoplamientos óptimos, donde el valor de α se encuentra optimizando para los dos espines de cada extremo. Un análisis más detallado de implementaciones previas de ambos tipos de cadenas puede encontrarse en [4] y sus referencias respectivas.

También se han implementado esquemas donde los acoplamientos en los extremos dependen del tiempo. Se mostró [16] que controlando únicamente dos espines en cada lado de la cadena y variando en el tiempo sus interacciones a primeros vecinos puede lograrse una fidelidad similar a la obtenida controlando todos o la mayoría de los espines en un sistema estático [2].

[2]Si bien este modelo no es independiente del tiempo, se incluye en esta categoría ya que se basa en control de interacciones y no en la aplicación de forzamientos externos sobre elementos individuales

1.4 Hamiltoniano de Heisenberg

Otro modelo independiente del tiempo que resulta de interés para este trabajo es el Hamiltoniano de Heisenberg ($\gamma = 0,\ \Delta = 1$ en (1.11)):

$$H_{Heis} = -\sum_{i=1}^{N-1} J_i(\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y + \sigma_i^z\sigma_{i+1}^z) \tag{1.22}$$

cuya expresión explicita en la base de una excitación puede encontrarse en el Apéndice A junto con la deducción de la misma.

Estudios sobre transmisión en las cadenas descriptas por este sistema han demostrado que la fidelidad es bastante pobre para acoplamientos homogéneos. Aún más, se ha probado que no es posible obtener transmisión perfecta con este modelo, incluso configurando los acoplamientos de forma individual [20]. De todas formas, sí es posible encontrar protocolos para lograr APST y PGST.

De forma análoga a lo que sucedía en el Hamiltoniano XX, la calidad de la transmisión tiene una fuerte dependencia del espectro de energías. Se ha estudiado [20] que pueden lograrse buenas transmisiones en espectros que satisfagan:

$$E_{i+1} - E_i \simeq q_i\alpha \tag{1.23}$$

con q_i siendo algún número impar. Es decir, puede lograrse PGST en cadenas cuyas energías difieran en múltiplos impares de una constante. En este caso, un problema de autovalores inverso resulta mucho más complejo que en el caso del Hamiltoniano XX. Sin embargo, es posible obtener conjuntos adecuados para los coeficientes de intercambio planteando la transmisión como un problema de optimización.

En particular, se toma como referencia el trabajo [11] cuyos autores, implementando un método propio de optimización, encontraron que puede obtenerse APST en el Hamiltoniano de Heisenberg mediante la configuración individual de los acoplamientos. Además, en un trabajo posterior [21], utilizando el mismo método, pudieron estudiar las propiedades de los espectros asociados. De este análisis, se concluyó que, un espectro "parcialmente ordenado" permite alcanzar APT, siempre y cuando, los autoestados asociados a esa porción del espectro estén localizados en los extremos de la cadena. Es decir, optimizando solamente algunos de los coeficientes de intercambio para lograr que una fracción de los autovalores cumpla la condición (1.23), puede lograrse una buena transmisión.

1.4.1 Heisenberg vs XX

Desde un punto de vista teórico, el Hamiltoniano de Heisenberg no resulta adecuado para la transmisión de estados. Las excitaciones a lo largo de cadenas de este tipo se propagan de forma desordenada.[3] Por otro lado, el Hamiltoniano XX, como se mostró

[3] Aquí la idea de "desorden" puede asociarse a oscilaciones rápidas en la localización de los estados. Un análisis cuantitativo del mismo, puede realizarse a través de la cantidad conocida como IPR. Algunos ejemplos de este tipo de caracterizaciones pueden encontrarse en [18], [21].

anteriormente tiene un espectro ordenado y permite encontrar transmisiones de alta fidelidad. Las diferencias analíticas entre ambos sistemas pueden entenderse en términos de su pertenencia a distintas clases de universalidad [22].

A pesar de las ventajas teóricas del modelo XX, desde un punto de vista experimental, los sistemas que se han logrado construir hasta ahora presentan un Hamiltoniano efectivo que tiende al de Heisenberg, o al menos con un parámetro de anisotropía no nulo ($\Delta \neq 0$). Luego, dada la creciente posibilidad de construir sistemas que manifiestan esa dinámica, el estudio de protocolos que logren una alta fidelidad en sistemas no isotrópicos resulta fundamental.

1.5 Modelo dependiente del tiempo

Otra clase de modelo que se plantea en este trabajo es un Hamiltoniano XX al que se añade control dependiente del tiempo a través de campos magnéticos. Este, puede escribirse como:

$$H^{dyn}(t) = -\frac{J}{2}\sum_{n=1}^{N-1}\left(\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y\right) + \sum_{n=1}^{N} B_n(t)\sigma_n^z \tag{1.24}$$

donde $B_n(t)$ representa los campos aplicados en cada miembro de la cadena. Es conveniente notar que en este caso el sistema ya no es estacionario. La aplicación de forzamientos externos variables en el tiempo para modificar el comportamiento en sistemas de espín se denomina control dinámico.

Control Dinámico y QSL

Debido al principio de incertidumbre de Heisenberg, existe un máximo teórico de la velocidad con la que la información puede transmitirse a lo largo de un sistema cuántico. Este se denomina "Límite Cuántico de Velocidad" y suele abreviarse como QSL por las siglas en inglés de *Quantum Speed Limit*. En el contexto de la transmisión de estados, el QSL implica que en cadenas homogéneas transmitir información de un extremo a otro de las mismas tomará un tiempo de orden $\mathcal{O}(N/2)$ [12]. Para sistemas no homogéneos, su valor depende fuertemente de los tipos de acoplamientos utilizados y el nivel de control en la cadena. La teoría de control óptimo (OCT) provee un método sistemático para calcular variacionalmente pulsos óptimos que permitan disminuir los tiempos de transmisión.

En el trabajo [12], se utiliza la OCT para estudiar el QSL controlando uno y luego dos espines en cada extremo de la cadena. Se muestra que los sistemas con QSL pueden controlarse si se consideran tiempos compatibles con ese límite. Es decir, mediante pulsos magnéticos optimizados puede lograrse una alta probabilidad de transmisión si se consideran tiempos de orden $\mathcal{O}(N/2)$. Otro ejemplo de control dinámico puede encontrarse en [23], donde se utilizan pulsos magnéticos de tipo parábolico, es decir, de la forma: $B_n(t) = C(t)[x_n - d(t)]^2$ con los valores de C y d encontrados mediante un algoritmo de optimización. A través de estos campos externos, se encuentra un tiempo mínimo de

transmisión dado por $t_{QSL} = (N-1)/(2J)$, correspondiente a la máxima velocidad de una onda de espín [19].

Capítulo 2

Algoritmos de optimización

Dado que la optimización es clave para una amplia variedad de disciplinas, el desarrollo de algoritmos para optimizar ha sido siempre una tarea importantísima. En las últimas décadas, los avances computacionales han permitido innovar fuertemente en el diseño e implementación de técnicas que resuelvan estos problemas.

2.1 Definición de un problema de optimización

Un problema de optimización puede definirse, a grandes rasgos, como la búsqueda de un conjunto de métricas asociadas a un sistema en un espacio definido por algún conjunto de restricciones. Matemáticamente, puede expresarse como la búsqueda de un punto $\boldsymbol{x} = [x_1, x_2, ..., x_n]$ que minimice alguna función objetivo f donde $\boldsymbol{x}$ pertenece a un subconjunto χ del dominio de f. A su vez, χ estará determinado por las restricciones del problema en cuestión [24].

Se define a los puntos $\mathbf{x}$ donde la función alcanza un mínimo (o máximo) como minimizadores (o maximizadores). Si se restringe el problema al caso de funciones diferenciables, los puntos de interés serán aquellos donde la derivada (o el gradiente en el caso multivariable) se anula. Estos puntos serán los candidatos a minimizadores o maximizadores de la función y se denominan puntos críticos. Dado que el problema de minimizar una función es equivalente al de maximizarla [24], se estudiarán únicamente los minimizadores. De todas formas, el análisis y la categorización de los puntos es directamente extensible a la maximización de una función.

En la figura 2.1, se muestran distintos tipos de puntos críticos. En primer lugar se observa un minimizador global donde f alcanza el mínimo valor de toda su imagen. Cabe destacar que aunque el mínimo global de f es por definición único, pueden existir distintos puntos $\mathbf{x}$ donde f alcance ese valor. Es decir, existe un único mínimo global pero puede existir más de un minimizador global. Al encontrar un candidato a mínimo suele ser difícil probar que es un mínimo global por lo que por lo general el interés radica en los mínimos y minimizadores locales. Dentro de los minimizadores locales, a su vez, es posible distinguir los fuertes y los débiles. Un minimizador local fuerte es un valor de $\mathbf{x}$ para el cual $f(\mathbf{x})$ es

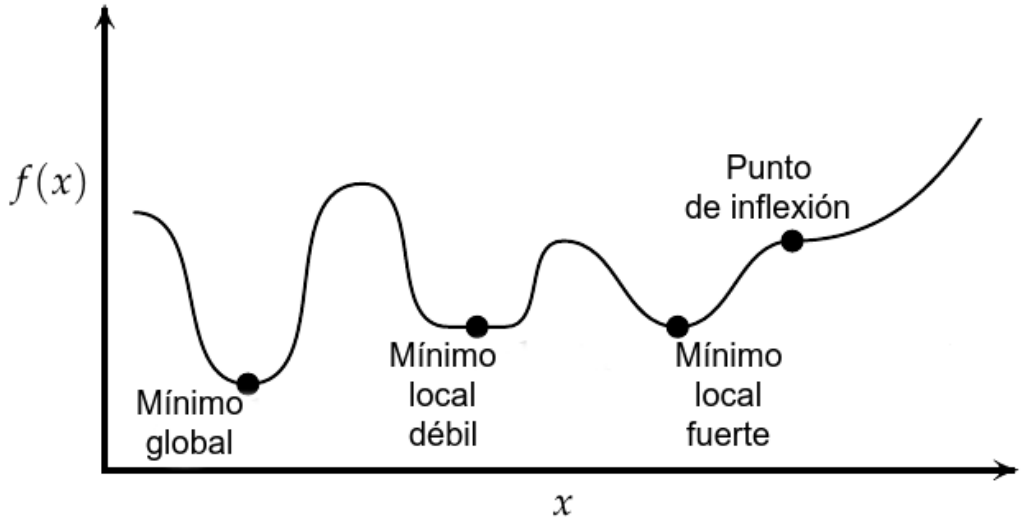

Figura 2.1: Gráfico de una función ilustrando los distintos tipos de puntos críticos.

menor a todos los puntos en un entorno dado. En términos más formales, puede definirse como un punto $\mathbf{x}^*$ para el que, dado un entorno $\delta > 0$, $f(\mathbf{x}^*) < f(\mathbf{x})$ para todo $\mathbf{x}^* \neq \mathbf{x}$ tal que $\|\mathbf{x} - \mathbf{x}^*\| < \delta$. Suele definirse a los mínimos locales débiles por oposición, es decir, serán aquellos que no sean fuertes. Un ejemplo de cada uno de ellos se observa también en la figura 2.1.

Otro tipo de punto crítico que es posible encontrar es un punto de inflexión o punto de silla. Estos son los puntos donde la derivada segunda cambia de signo pero la función no alcanza un extremo. Los puntos de inflexión se corresponden con un extremo de la derivada primera. Es importante destacar de esta clasificación que una derivada nula es una condición necesaria pero no suficiente para que la función alcance un extremo.

Para el caso de funciones diferenciables, la condición de derivada primera nula ($\nabla f(\mathbf{x})$) se conoce, en el contexto de la optimización de funciones de varias variables, como condición necesaria de primer orden (o FONC, por sus siglas en inglés: *first order necessary condition*). Para que $\mathbf{x}$ sea un minimizador local, es también necesario que la derivada segunda de la función en el punto sea definida semi-positiva ($\nabla^2 f(\mathbf{x}) \geq 0$). Esta condición, se denomina condición necesaria de segundo orden (o SONC: *second order necessary condition*). Para la optimización de una función doblemente diferenciable no restringida, ambas combinadas son necesarias pero no suficientes. Luego, se define la condición suficiente de segundo orden (SOSC: *second order sufficient condition*) que asegura que el punto $\mathbf{x}$ es un minimizador local fuerte si se satisface la FONC y $\nabla^2 f(\mathbf{x})$ es definida positiva (estrictamente mayor a cero).

Cabe destacar que aunque se expliciten estas condiciones para problemas diferenciables, las definiciones de puntos críticos son válidas también para funciones no diferenciables. Más aún, muchos de los métodos descriptos a continuación no dependen del gradiente y resultan convenientes para la optimización de ese tipo de funciones.

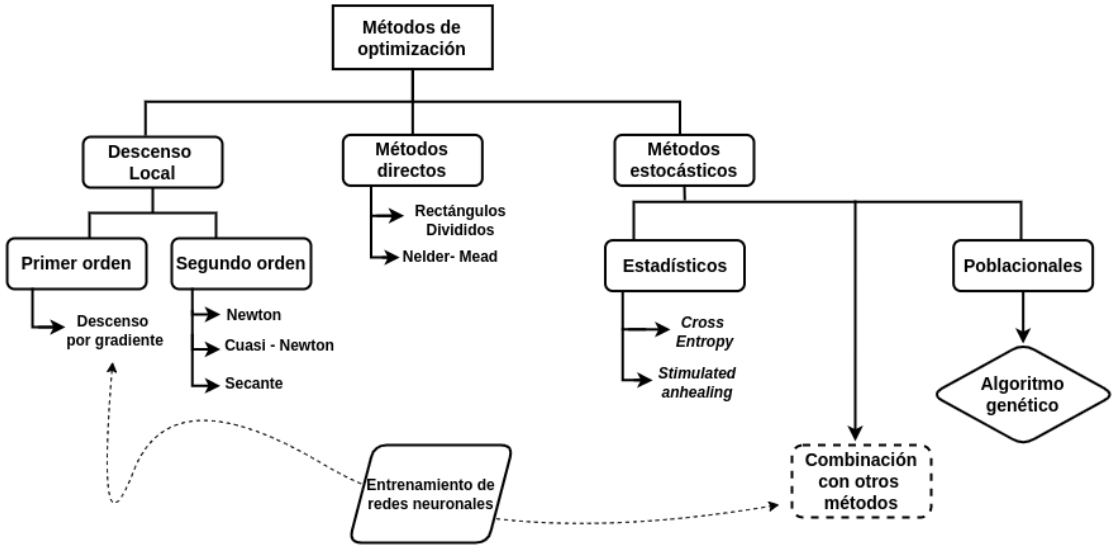

Figura 2.2: Clasificación general de los métodos de optimización descriptos. Se destacan el algoritmo genético y la utilización de métodos estocásticos combinados con descenso por gradiente dada su relevancia en el trabajo.

2.2 Clasificación de algoritmos de optimización

Debido a la extensa cantidad de estrategias que se han desarrollado para resolver problemas de optimización en varias variables, es conveniente introducir una clasificación general de los métodos más utilizados. Un resumen de los métodos que se mencionan a continuación se esquematiza en la figura 2.2. Una descripción más completa de cada uno de ellos puede consultarse en [24].

2.2.1 Descenso local

Un enfoque general muy utilizado es el descenso local que usa información de las derivadas de la función a minimizar. Este tipo de métodos consisten principalmente en elegir una dirección de descenso e ir tomando pasos en esa dirección hasta alcanzar alguna condición de convergencia. La dirección de descenso puede elegirse de distintas formas. Dentro de las más típicas se encuentran los métodos de primer orden, que utilizan información provista por el gradiente de la función. A su vez, uno de los métodos más populares en esta categoría es el descenso por gradiente.

Descenso por gradiente

El método de descenso por gradiente se basa en la idea intuitiva de que una dirección óptima para acercarse al mínimo de la función será aquella dada por la mayor pendiente de descenso. Esta estará dada por el opuesto del gradiente $\nabla f(\mathbf{x})$. Si se considera un algoritmo de minimización, donde $\mathbf{x}^{(k)}$ es el punto que se tiene en la k-ésima iteración, se define el gradiente según:

$$\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)}) \tag{2.1}$$

De esta forma, la dirección de mayor pendiente de descenso en la k-ésima iteración estará dada por:

$$\mathbf{d}^{(k)} = -\frac{\mathbf{g}^{(k)}}{\|\mathbf{g}^{(k)}\|} \tag{2.2}$$

Entonces, se parte de algún candidato a minimizador $\mathbf{x}^{(0)}$ y se va actualizando su valor en la dirección $\mathbf{d}^{(k)}$ según:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)} \tag{2.3}$$

donde α se conoce como tasa de aprendizaje y es un parámetro que define la rapidez con la que se actualizan los valores. Esto se repite hasta alcanzar algún criterio de convergencia.

Existen distintas modificaciones que pueden aplicarse a los métodos de descenso por gradiente para mejorar su desempeño. Otra opción es utilizarlo en combinación con otras técnicas para que su implementación sea más eficiente. Un ejemplo particular de esto es su utilización en los algoritmos de aprendizaje automático que se discute más adelante.

Por otro lado, se encuentran los métodos de segundo orden que utilizan información relacionada con el Hessiano (derivadas segundas). Entre estos se hallan los métodos de Newton, Cuasi-Newton y secante.

2.2.2 Métodos directos

Otro acercamiento posible es la utilización de métodos directos. Los mismos no utilizan información relacionada a las derivadas sino otros criterios. Por ejemplo, el descenso por coordenada y el método de Powell que optimizan sobre distintas direcciones. O los métodos Hooke-Jeeves y de búsqueda por patrones generalizados (que adaptan el tamaño de los pasos en base a diferentes criterios). El método de Nelder-Mead y el algoritmo de rectángulos divididos también pertenecen a esta categoría [24].

2.2.3 Métodos estocásticos

Un tercer enfoque es la implementación de aleatoriedad en la exploración del espacio, en esto se basan los métodos estocásticos. Muchos de estos métodos resultan de añadir aleatoriedad a los mencionados anteriormente evitando así que los algoritmos queden "atrapados" en mínimos locales. Por ejemplo, se utiliza un método aleatorio para elegir las direcciones de optimización en el caso de los métodos directos. Otra estrategia común es combinarlos con el descenso por gradiente. En particular, el descenso estocástico por gradiente es una forma típica de entrenamiento de redes neuronales. También se incluyen en esta categoría métodos de optimización inspirados en la mecánica estadística como los métodos de "*Cross-Entropy*" y *"Simulated Annealing"*.

Ejemplo de algoritmo combinado.

En una etapa de exploración del problema de transmisión de estados, se reprodujeron los resultados de la parte inicial del trabajo [19]. En él, se aplica un algoritmo simple de aprendizaje automático que resulta de combinar la técnica de descenso por gradiente con un generador de números aleatorios. A continuación, se describe su implementación a modo de ejemplo.

Para comenzar se inicializan los acoplamientos con algún criterio o de forma aleatoria. En particular, siguiendo el trabajo de referencia, para la cadena de N espines se parte de los de la de N-1, comenzando con una cadena de 3 espines inicializada con todos sus J_i iguales a 1. Luego, se repite la siguiente secuencia hasta alcanzar la tolerancia o el número máximo de iteraciones:

1. Se elige un entero m aleatorio entre 1 y N, que corresponde al sitio del spin que se perturba.

2. Se calculan las cadenas de espín perturbadas $J^{N\pm}_{i,i+1} = J^{N}_{i,i+1} \pm \beta\delta_{im}$

3. Se calcula el gradiente a partir de la función de pérdida evaluada en ambas cadenas perturbadas:
$$\mathbf{g_m} = \frac{L\left(J^{N+}\right) - L\left(J^{N-}\right)}{2\beta} \tag{2.4}$$
donde $L(\mathbf{J^N})$ es una función costo asociada a la fidelidad y eficiencia en la transmisión.

4. Una vez calculado el gradiente asociado a la perturbación se actualiza el i-ésimo acoplamiento $J^{N}_{m,m+1} \leftarrow J^{N}_{m,m+1} - \alpha g_m$

5. Finalmente, se calcula la fidelidad con los acoplamientos actualizados y se verifica si cumple con la tolerancia pedida. Una vez alcanzada la tolerancia o el número máximo de iteraciones se corta el ciclo y se almacenan los acoplamientos.

Se implementó este algoritmo en primer lugar para el Hamiltoniano XX, y luego para el Hamiltoniano de Heisenberg, tomando cadenas de largos entre 3 y 30 y una tolerancia de 0.0001. Los resultados obtenidos para la fidelidad máxima optimizando sobre los acoplamientos se muestran en la figura 2.3.

Sin entrar en detalles sobre este algoritmo en particular ni las diferencias entre ambos sistemas físicos, pueden extraerse dos conclusiones generales:

- Al aumentar las dimensiones del problema, en general, encontrar valores óptimos resulta más complicado.

- Un algoritmo puede comportarse de manera estable y proveer buenos resultados para algunos sistemas pero no hacerlo para otros.

Además, este método ilustra de forma simple cómo puede emplearse un descenso estocástico por gradiente para entrenar un algoritmo de aprendizaje automático.

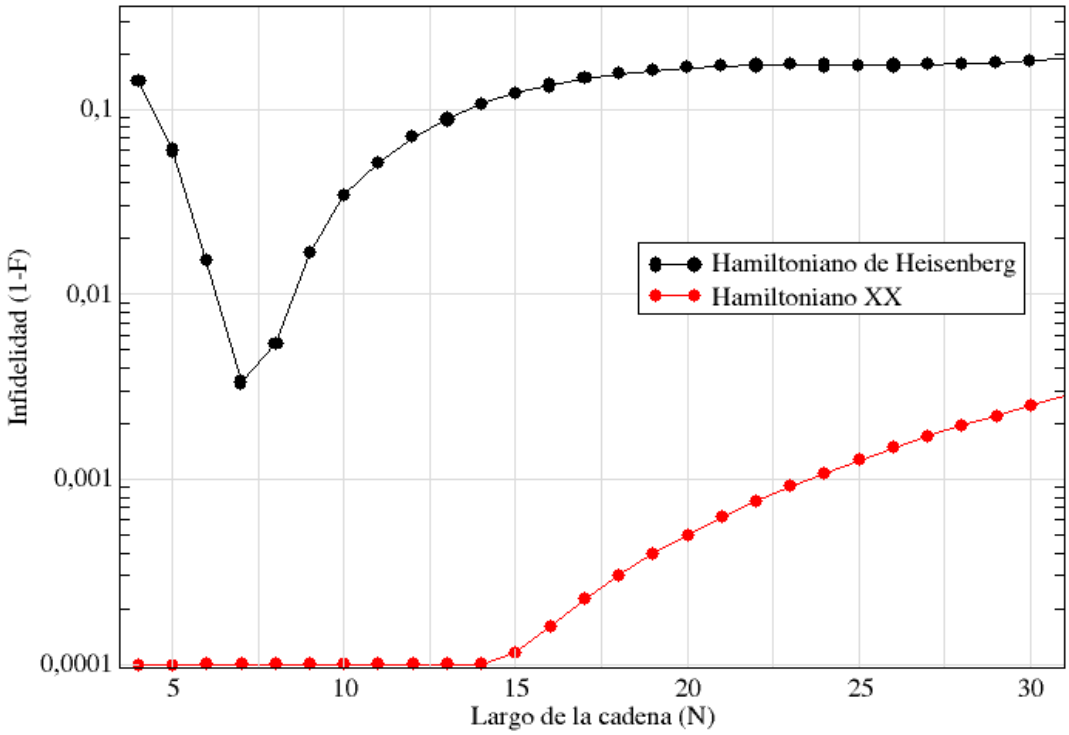

Figura 2.3: Valores de infidelidad mínima alcanzada utilizando el algoritmo descripto para el Hamiltoniano XX (rojo) y el Hamiltoniano de Heisenberg (negro). Se realizaron un millón de iteraciones del algoritmo para cada dimensión.

Métodos Poblacionales

Una sub-categoría dentro de los métodos estocásticos son los métodos de población. Estos últimos se basan en utilizar una gran colección de conjuntos de puntos distribuidos de forma generalmente aleatoria en el espacio de búsqueda y, a través de evaluaciones de la función en cada uno de ellos, encontrar cuales se corresponden con un minimizador (o maximizador). Es aquí donde se sitúan los algoritmos genéticos que, dada su relevancia en el presente trabajo, se describen en detalle en la siguiente sección.

2.3 Algoritmo genético

El algoritmo genético (GA) es un algoritmo estocástico de población que utiliza operadores de "selección", "cruza" y "mutación" inspirados en la teoría Darwiniana [25]. Se basa en la idea de que los individuos más aptos serán los que "pasen" sus genes a la siguiente generación. En el marco de un problema de optimización, se define la aptitud como la función que se busca maximizar (o la inversa de la que se busca minimizar). El esquema general para su implementación se muestra en la figura 2.4.

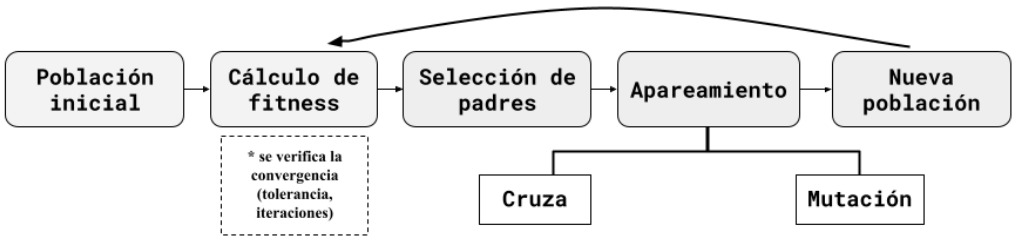

Figura 2.4: Esquema general del funcionamiento del algoritmo genético.

Para comenzar, se propone una población inicial, en principio, aleatoria. Es decir, se genera un conjunto conformado por distintos vectores, $\mathbf{X} = (x_1, x_2, ...)$, candidatos a ser el maximizador de la función aptitud. De acuerdo al problema que se desee resolver, pueden generarse los distintos vectores en la población usando distribuciones uniformes, gaussianas, o que respeten algún tipo de regla que aplique al sistema particular. Cada posible solución, es decir, cada vector $\mathbf{X}$ en la población, se denomina "individuo"[1] y representa una solución posible. A su vez, cada uno de los parámetros (x_k) correspondientes al individuo se denominan "genes".

En cada iteración del algoritmo, denominada generación, se calcula la aptitud asociada a cada uno de los individuos. Luego, se seleccionan los individuos "padres", es decir, aquellos que darán origen a la siguiente generación. Sobre este conjunto de individuos, se realizan las operaciones de *crossover* o cruza y mutación. La cruza combina genes presentes en dos "padres" para generar un miembro de la nueva población. Por otro lado, la mutación realiza cambios sobre los genes en un mismo individuo. Existen distintos métodos para realizar las operaciones de selección, cruza y mutación, algunos de ellos, se describen en la sección 2.3.1.

En este trabajo para implementar algoritmos genéticos se utiliza PyGAD [26], una librería open-source de Python desarrollada específicamente para este propósito.

2.3.1 Operadores en un algoritmo genético

A continuación, se describen algunos de los operadores típicos de cruza, mutación y selección. Estos están integrados por defecto en PyGAD. También es posible ingresar a la librería funciones o métodos personalizados para cada operador.

Selección de padres

Como se describió previamente, en la fase de selección de padres, se eligen los individuos que darán origen a la siguiente generación. De manera análoga al proceso de selección natural, se espera que el algoritmo favorezca la reproducción de los individuos más aptos.

[1] Según la literatura o documentación, también pueden encontrarse como cromosomas

Algunos de los métodos con los que puede modelarse esta elección son:

- Ruleta (*RWS: roulette wheel selection*): Asigna probabilidades ($p(i)$) a los distintos individuos de ser elegidos de acuerdo a su aptitud ($f(i)$). Es posible ilustrar este método pensando en una ruleta donde la porción asociada a cada individuo es proporcional a su aptitud. Para elegir cada padre, se hace girar la ruleta. La probabilidad de que el i-ésimo individuo sea elegido está dada por:

$$p(i) = \frac{f(i)}{\sum_{j=1} f(j)} \tag{2.5}$$

  Los individuos más aptos tienen más posibilidades de ser elegidos, sin embargo, todos pueden ser tomados como padres. Este mecanismo de selección es útil para evitar máximos locales ya que habilita la aparición de soluciones diversas [25].

- Estacionaria (*SSS: steady state selection*): En cada generación se seleccionan los más aptos y luego se reemplazan los menos aptos con la nueva generación. Este mecanismo de selección preserva una mayor parte de la generación anterior ya que solo renueva unos pocos miembros.

- Ranking/Clasificación: Se ordena a los individuos de acuerdo a su aptitud. Luego, se le asigna a cada uno una nueva aptitud acorde. El peor tendrá aptitud igual a 1 y el mejor tendrá aptitud igual al número total de individuos. Finalmente, implementa *RWS* pero utilizando probabilidades basadas en esta nueva asignación. De esta forma, ya no importan las probabilidades relativas. Puede prevenir la convergencia rápida.

- Torneo: Se realizan varios torneos eligiendo k individuos aleatoriamente. El ganador de cada uno de los "torneos" es seleccionado como padre. Es decir, se arman tantos grupos (torneos) de individuos como padres deseen seleccionarse y en cada uno se selecciona como ganador al más apto del grupo. La "presión de selección" es la probabilidad que tiene un individuo de ser seleccionado en un torneo. Esta aumenta junto con el número k de individuos por grupo. De esta forma, un k mayor implica menos probabilidad de selección de los individuos menos aptos, ya que hay más probabilidad de que un individuo más apto pertenezca al mismo grupo en que compite.

- Estocástica universal (*SUS: stochastic universal selection*): Es similar al mecanismo *RWS* con la diferencia de que en lugar de girar la ruleta para cada padre, esta se hace girar una única vez y se seleccionan individuos equiespaciados respecto al punto elegido.

- Selección completamente aleatoria.

Cruza o *crossover*

A partir de los individuos seleccionados, se crea la siguiente generación a través de la cruza. Las funciones *crossover* predefinidas en PyGAD (esquematizadas en la figura 2.5) son:

- Punto único: toma los individuos (A y B) seleccionados y define dos nuevos eligiendo un gen al azar e intercambiando todos los genes a partir de ese.
- Dos puntos: se eligen dos puntos aleatorios en los individuos, y se generan los nuevos intercambiando los genes contenidos entre esos dos puntos.
- Uniforme o "Punto a Punto": Para cada gen se elige aleatoriamente entre alguno de los padres.

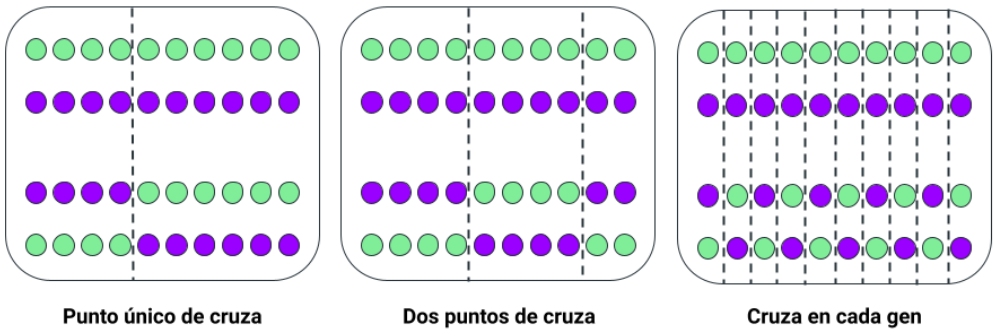

Figura 2.5: Esquema de los distintos métodos de cruza. De izquierda a derecha se muestran los métodos de: cruza en un único punto, cruza en dos puntos y cruza uniforme o punto a punto.

Mutación

En cada generación también es posible que algunos de los individuos en la población sufran una mutación. Las funciones mutación provistas por PyGAD son:

- Aleatoria: Se suma un valor aleatorio dentro de un rango fijo a cada gen.
- Intercambio o *swap*: Se intercambian dos genes.
- Inversión: Se invierte el orden de una secuencia de genes
- Mezcla o *scramble*: Reordena todos los genes aleatoriamente.

Como se mencionó anteriormente también es posible definir funciones de mutación, cruza y selección personalizadas. Además, es posible elegir la probabilidad con la que queremos que ocurran los procesos de cruza y de mutación. Para el caso de la mutación, también es posible definir una mutación adaptativa que es de tipo aleatoria pero que permite ingresar como parámetro dos probabilidades distintas según si las fidelidades están cercanas o no al valor promedio.

Capítulo 3

Aprendizaje por refuerzos

Dentro del aprendizaje automático, es posible distinguir distintos paradigmas de aprendizaje entre los que se suelen mencionar el supervisado y el no supervisado. En el aprendizaje supervisado, el entrenamiento se realiza con datos etiquetados. Es decir, se cuenta con un conjunto de datos y cada uno de ellos se corresponde con una etiqueta. Estos se utilizan para luego poder hacer predicciones, generalmente de categorías (clasificación) o de valores continuos (regresión), sobre datos no etiquetados. Por otro lado, el aprendizaje no supervisado busca encontrar patrones y estructuras en conjuntos de datos inicialmente no etiquetados [27].

Este trabajo se concentra en un tercer tipo de aprendizaje automático denominado aprendizaje por refuerzos o *Reinforcement Learning* (RL). Este paradigma se basa en aprender a través de un agente activo que interactúa con el ambiente a través de acciones. El agente percibe el estado del ambiente antes y después de realizar una acción y así puede aprender de sus experiencias pasadas a través de la obtención de recompensas.

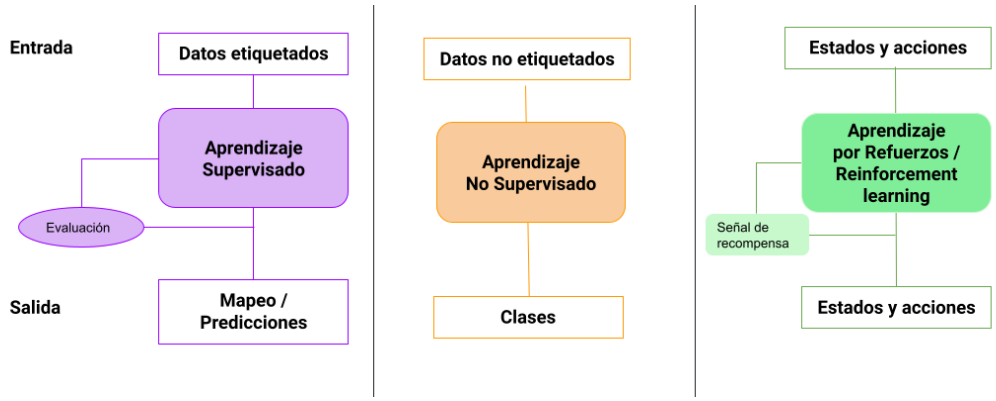

Figura 3.1: Comparación de los paradigmas de aprendizaje automático. A la izquierda, el aprendizaje supervisado que tiene como entrada datos con sus etiquetas y luego de una evaluación sobre ellos permite mapear o realizar predicciones sobre datos nuevos. En el centro, el aprendizaje no supervisado que permite clasificar datos no etiquetados. Finalmente, el aprendizaje por refuerzos que recibe estados y acciones, y decide los próximos a partir de señales de recompensa.

Al resolver un problema utilizando RL es necesario definir un conjunto de estados posibles del sistema que se está estudiando, esto se conoce como espacio de estados ($\mathcal{S}$). Cada estado provee una caracterización completa del ambiente. En este trabajo, se consideran únicamente espacios de estados discretos pero también los hay de tipo continuo. También, es fundamental especificar el conjunto de acciones posibles que puede tomar el agente para modificar el estado del ambiente. Estas conforman el espacio de acciones ($\mathcal{A}$).

Una visión general de la implementación de este tipo de algoritmos es la siguiente: Para un paso temporal t el agente registra el estado del sistema $s_t \in \mathcal{S}$ y, a partir de esa observación, decide tomar la acción $a_t \in \mathcal{A}$. El ambiente E recibe esa acción y el estado s_t se actualiza al s_{t+1}. Luego, el agente obtiene una recompensa r_t como consecuencia de este proceso. Finalmente, almacena la misma junto con la acción tomada y el estado anterior y posterior a realizarla. Esto conforma una *experiencia* que luego el agente podrá usar para elegir sus acciones futuras. Coloquialmente, podría decirse que en este caso el aprendizaje se produce por "prueba y error" [28].

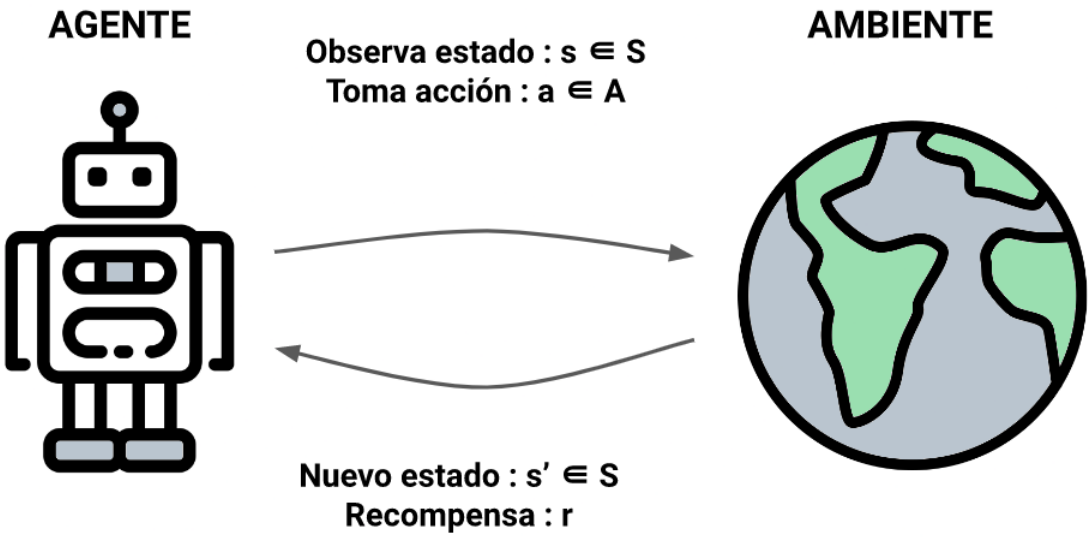

Figura 3.2: Esquema general de un algoritmo de aprendizaje por refuerzos. El agente observa el estado del ambiente y en función de eso realiza una acción. Se actualiza el estado según la acción que tomó y el agente recibe una recompensa acorde.

3.1 Elementos de un algoritmo de aprendizaje por refuerzos.

Además del agente, el ambiente y los espacios de estados y acciones, existen cuatro elementos fundamentales en un algoritmo de aprendizaje por refuerzos. Estos son: una política, una recompensa, una función de valor y opcionalmente un modelo del ambiente [29].

3.1.1 Política

La política es un conjunto de reglas que describe cómo el agente toma sus decisiones. Matemáticamente, es una función que mapea un estado s_t a una acción a_t:

$$a_t = \pi(s_t) \tag{3.1}$$

De esta forma, dado el estado del ambiente en ese tiempo, el agente decide qué acción tomar a continuación. La política puede ser una tabla o función en algunos casos simples. En otros casos más complejos puede utilizarse algún tipo de cálculo computacional más extensivo.

Política "ϵ-*greedy*".

Una política posible para que el aprendizaje realmente sea por "prueba y error" sería que el agente tome decisiones completamente aleatorias. Esta opción, posiblemente, conllevaría una gran cantidad de episodios hasta que el agente aprenda qué secuencia de acciones es la más conveniente. En el otro extremo, se encuentra la posibilidad de repetir únicamente las acciones que le dieron una alta recompensa. Sin embargo, restringiéndose a las mismas, podría dejar sin explorar acciones que provean aún más.

La política de toma de decisiones que suele implementarse se denomina "ϵ-*greedy*". Según esta, el agente elige con probabilidad ϵ una acción aleatoria, en caso contrario elige la acción que, de acuerdo a sus experiencias previas, le proveerá mayor recompensa a futuro[1]. El término "greedy", que significa avaro en inglés, representa el balance que hace el agente entre elegir acciones que "sabe" que dan recompensa y explorar acciones distintas, dada la posibilidad de que le den aún más.

Lo ideal es elegir un valor de ϵ que varíe a medida que se realizan más episodios. Al comienzo del aprendizaje, es conveniente que el agente explore la mayor cantidad de acciones posibles. Esto se corresponde con un valor alto. A medida que avanza, se toma un ϵ cada vez menor. Así, el agente basa las decisiones cada vez más en su experiencia a medida que acumula más aprendizaje.

3.1.2 Recompensa

La recompensa le indica al agente qué eventos (pares estado-acción) son exitosos. En cada paso temporal, el ambiente devuelve al agente una señal o recompensa instantánea r_t. El objetivo del agente será maximizar la recompensa total que recibe a largo plazo. Matemáticamente, la recompensa total R obtenida hasta el paso temporal t puede escribirse como:

$$R = r_0 + \gamma^1 r_1 + ... + \gamma^t r_t \tag{3.2}$$

[1]Se verá más adelante que esto equivale a decir que elige la acción con mayor valor Q ($a_t = \max_a Q_(s_t, a, W)$).

donde γ se conoce como factor de descuento y toma valores entre 0 y 1. Su función es que mientras más pasos temporales tarde el agente en llegar a una meta, menos recompensa reciba. De esta forma, prioriza un aprendizaje más rápido.

3.1.3 Función valor

La función valor especifica qué es conveniente a largo plazo. En lugar de ser una apreciación instantánea como la recompensa, la función valor es la cantidad total de recompensa que el agente puede esperar acumular en el futuro. De esta forma, un estado puede dar una baja recompensa inmediata pero al estar seguido de estados que provean mayor recompensa corresponderse con un valor alto.

Función valor de estado

La función valor de estado V toma un estado s y una política π y devuelve un valor escalar que representa la recompensa acumulada esperada al llegar al paso temporal t:

$$V_\pi(s) = \mathcal{E}_\pi[R_t|s_t = s] = \mathcal{E}_\pi[R = r_0 + \gamma^1 r_1 + ... + \gamma^t r_t|s_t = s] \tag{3.3}$$

De esta forma, V permite predecir qué tan bien una secuencia de acciones funcionará incluso antes de que el agente las realice.

Función de valor estado-acción

Esta función también denominada función Q especifica qué tan bueno es para el agente realizar una acción particular (a) para un estado (s) bajo una política (π). Se define según:

$$Q_\pi(s, a) = \mathcal{E}_\pi[R_t|s_t = s, a_t = a] = \mathcal{E}_\pi[R = r_0 + \gamma^1 r_1 + ... + \gamma^t r_t|s_t = s, a_t = a] \tag{3.4}$$

La diferencia entre ambas funciones de valor es que V establece qué tan bueno es un estado determinado mientras que Q establece qué tan bueno o conveniente es realizar una acción estando en un estado. Ambas funciones suelen representarse en una tabla que muestra la correspondencia entre un estado (para V) o un par estado-acción (para Q) y su valor [30].

3.1.4 Modelo

El modelo funciona imitando el comportamiento del ambiente y provee al agente un entendimiento general de cómo funciona. El agente puede utilizar el modelo para predecir cómo reaccionará el ambiente a determinada acción para un estado dado. Puede expresarse como una función M que predice la probabilidad del próximo estado dados el estado actual y una acción sobre el mismo:

$$M(s_t, a_t) = p(s_{t+1}|s_t, a_t) \tag{3.5}$$

3.2 Fundamentos Matemáticos

3.2.1 Ecuaciones de Bellman

Para resolver un problema de RL, se busca encontrar una política que logre una alta recompensa final. Las funciones de valor permiten ordenar o clasificar las políticas, es decir, se dice que π es mejor que π' si y solo si $V_\pi(s) \geq V_{\pi'}(s)$ para cualquier estado s. Siempre existe al menos una política que es mejor o igual a todas las demás y se denomina política óptima.

De acuerdo a la notación de [29], se representa a todas las políticas óptimas con π_*. Todas comparten la misma función óptima de valor de estado (v_*) y de estado-acción (Q_*). Es decir,

$$v_*(s) = \max_\pi v_\pi(s) \tag{3.6}$$

$$Q_*(s) = \max_\pi Q_\pi(s, a) \tag{3.7}$$

para todo estado s y acción a.

Las funciones óptimas, a diferencia de las políticas óptimas, son únicas para un problema dado. Las ecuaciones de Bellman son condiciones especiales de consistencia que permiten encontrar las funciones de valor óptimo y consecuentemente una política óptima. Estas pueden escribirse como:

$$Q_*(s) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \max_{a'} Q_*(s', a') \right] \tag{3.8}$$

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \tag{3.9}$$

Las ecuaciones (3.8) y (3.9) son en realidad sistemas de ecuaciones, una para cada estado posible del sistema. Luego, si se cuenta con un modelo que representa la dinámica del sistema, es posible, resolverlas para encontrar las funciones óptimas de valor [29].

3.2.2 Proceso de decisiones de Markov (MDP)

Los procesos de decisiones de Markov proveen un marco matemático para la mayoría de los problemas de aprendizaje de refuerzo [30]. El agente toma decisiones en función de lo que percibe del estado del ambiente. Idealmente, el agente debería recibir una señal que resuma de alguna forma la historia del sistema. Este tipo de señales que retienen toda la información relevante se denominan Markovianas y se dice que cumplen con la propiedad de Markov. La propiedad de Markov establece que un estado futuro depende únicamente del presente y no del pasado. Por ejemplo, la velocidad y posición actual de un proyectil determinan completamente su trayectoria futura (sin importar cómo la haya alcanzado).

En el contexto de un problema de RL, es posible formular la propiedad Markoviana

como sigue. Se considera un número finito de estados y recompensas[2]. La acción con la que el ambiente responde para tiempo $t+1$ a la acción A_t que se realizó en el tiempo t estará dada por la dinámica del sistema. Entonces, puede definirse la probabilidad de que ocurra un estado futuro posible s' con recompensa r como:

$$P_r\{R_{t+1} = r, S_{t+1} = s' | S_0, A_0, R_1, ..., S_{t-1}, A_{t-1}, R_t, S_t, A_t\} \tag{3.10}$$

donde $S_0, A_0, R_1, ..., S_{t-1}, A_{t-1}, R_t, S_t, A_t$ son los valores de eventos pasados. Si la señal de estado tiene la propiedad de Markov, entonces la respuesta en $t+1$ depende solamente de los eventos en t. Entonces la esta probabilidad puede escribirse como:

$$P_r\{R_{t+1} = r, S_{t+1} = s' | S_t, A_t\} \tag{3.11}$$

para todo r,s', S_t y A_t [29].

Una cadena Markoviana es un modelo probabilístico en el que cada evento depende únicamente del anterior. Los procesos de decisiones de Markov (MDP) extienden la idea de la cadena agregando la capacidad de un agente de tomar decisiones.

3.3 Deep Reinforcement learning y redes neuronales

El Deep Learning (DL) constituye un área del aprendizaje automático que se caracteriza por el uso de redes neuronales para representar los problemas [30]. A su vez, el aprendizaje en un modelo de DL puede ser supervisado, no supervisado o, como en el caso de estudio, por refuerzos. Este último caso que combina redes neuronales con aprendizaje por refuerzos se conoce como *Deep Reinforcement Learning* o DRL.

Una red neuronal puede pensarse como una entidad matemática que logra representar funciones complicadas a través de la composición de funciones más simples. El primer modelo de redes neuronales fue propuesto por Frank Rosenblatt en 1958, inspirado por el funcionamiento del sistema nervioso [31]. Rosenblatt propuso una red de interacciones denominada perceptrón que modelaba la conexión entre neuronas biológicas. Estas neuronas artificiales computan una combinación lineal de algún vector provisto como entrada. Además, devuelven una transformación de esta combinación lineal denominada activación, dada por una función no lineal. Formalmente, si se supone que una neurona artificial recibe un vector de entrada $\boldsymbol{x} \in \mathcal{R}^n$, lo transformará según:

$$a(\boldsymbol{x}) = f\left(\sum_i^n w_i x_i + b\right) \tag{3.12}$$

donde $w = (w_1, w_2, ..., w_n)$ son los pesos de la red, b es una constante que representa el sesgo y f es alguna función no lineal de activación. Algunas de las funciones típicamente utilizadas se describen en la sección 3.3.1.

En una red neuronal típica, se tienen múltiples capas de neuronas artificiales (que

[2]Esto permite simplificar el análisis pero la explicación es extensible a sistemas continuos.

desde este punto en adelante se denominaran simplemente neuronas). Cada neurona en una capa, está conectada a todas las de la capa anterior y siguiente, componiendo lo que se conoce como capas completamente conectadas o *"fully connected layers"*. Es importante destacar que las neuronas en una misma capa no interactúan entre ellas.

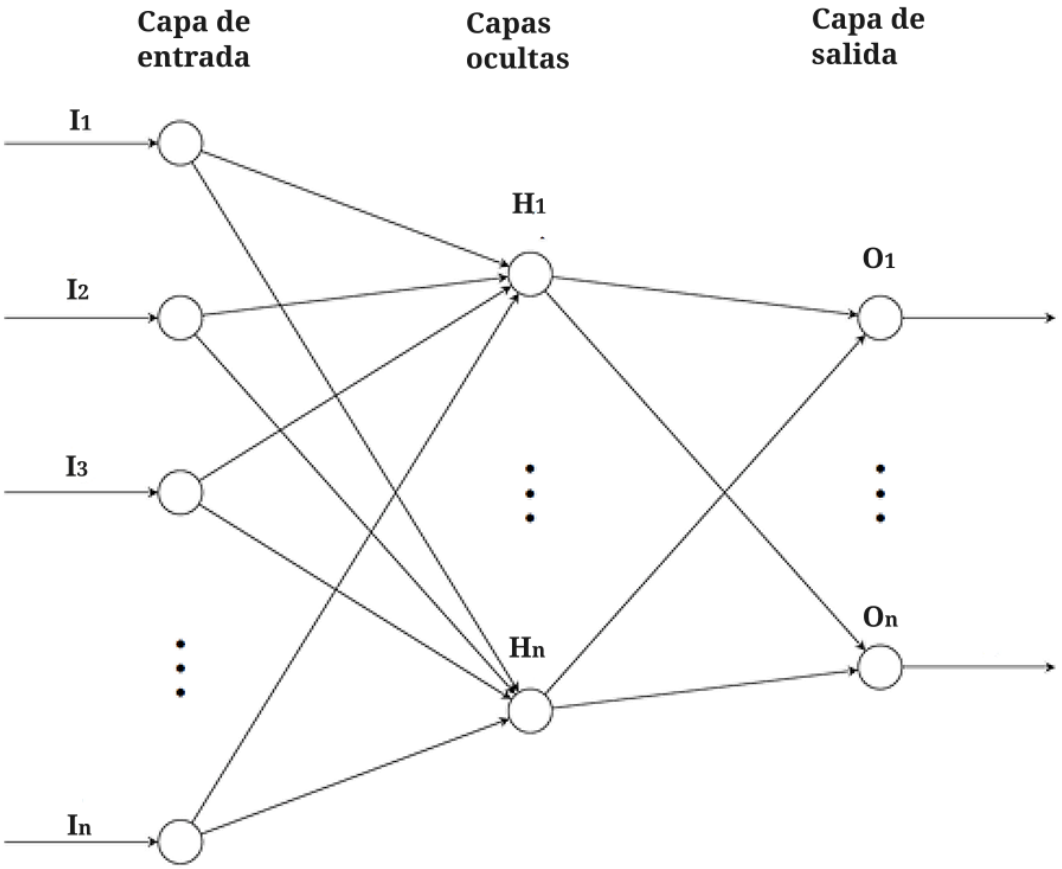

Figura 3.3: Esquema típico de la arquitectura de una red neuronal. Se distinguen tres componentes principales: la capa de entrada, las capas ocultas y la capa de salida [28].

Los pesos asociados a cada capa l se representan con una matriz W^l:

$$W^l = \begin{bmatrix} w^l_{11} & w^l_{12} & w^l_{13} & ... & w^l_{1n} \\ w^l_{21} & w^l_{22} & w^l_{23} & ... & w^l_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w^l_{d1} & w^l_{d2} & w^l_{d3} & ... & w^l_{dn} \end{bmatrix} \tag{3.13}$$

donde cada elemento w^l_{ij} denota el peso de la conexión entre la i-ésima neurona de la capa anterior y la j-ésima de la capa l. Las dimensiones de la matriz de peso W^l serán $n \times d$ donde n es el número de neuronas en la capa l y d el número de neuronas en la capa

siguiente. Sea $B^l = (b_1, b_2, ..., b_n)$ un vector que contiene el sesgo asociado a cada neurona de la capa l, puede definirse la activación a^l de esa capa como:

$$a^l(\boldsymbol{x}) = f(W^l a^{l-1}(\boldsymbol{x}) + B^l), \tag{3.14}$$
$$a^0(\boldsymbol{x}) = \boldsymbol{x}$$

donde $a^0(\boldsymbol{x})$ representa el vector de entrada de la red. El vector de salida de la red, suponiendo que se tienen n capas, se denota con $\hat{y}$ y está dado por:

$$\hat{y} = f(W^n a^{n-1}(\boldsymbol{x}) + B^n) \tag{3.15}$$

Este tipo de redes neuronales con varias capas son comúnmente denominadas perceptrones de múltiples capas (MLP, por sus siglas en inglés). Un esquema básico de la estructura de un MLP se muestra en la figura 3.3. En primer lugar, los datos ingresan a través de la capa de entrada. Esta capa tendrá tantas neuronas como datos de entrada tenga el sistema. Luego, se tienen las capas ocultas. Se llama así cualquier capa entre la capa de entrada y la capa de salida. En ellas, los datos son transformados usando una serie de funciones lineales y no lineales. El número de capas ocultas depende de la complejidad del sistema a tratar. Finalmente, la capa de salida devuelve los datos ingresados convertidos en una decisión o predicción.

3.3.1 Funciones de activación

Las funciones de activación son las responsables de la no linealidad en las redes. Algunos de los tipos más comunes de funciones de activación usadas son:

- **Función Sigmoide**: Se define como $f(z) = 1/(1+e^{-z})$. Esta función re-escalea los valores de z en el rango de 0 a 1. También es llamada función logística.
- **Tagente hiperbólica**: Se define como $f(z) = (e^{2z}-1)/(e^{2z}+1)$. Como se muestra en la figura 3.4, tiene una forma similar a la función sigmoide pero está centrada en el origen proveyendo valores entre -1 y 1.
- **Función ReLU o Función Lineal Rectificadora Unitaria**: Es una de las funciones de activación más utilizadas. Toma el valor 0 cuando z es menor a 0 y z cuando z es mayor o igual a 0. Formalmente, puede expresarse como $f(z) = \max(0, z)$.
- **Función Softmax**: Es una generalización de la función Sigmoide [30]. Suele utilizarse en problemas de clasificación ya que provee las probabilidades de que un elemento pertenezca a cada clase. Puede definirse como: $f_i(z) = e^{z_i} / \sum_j e^{z_j}$

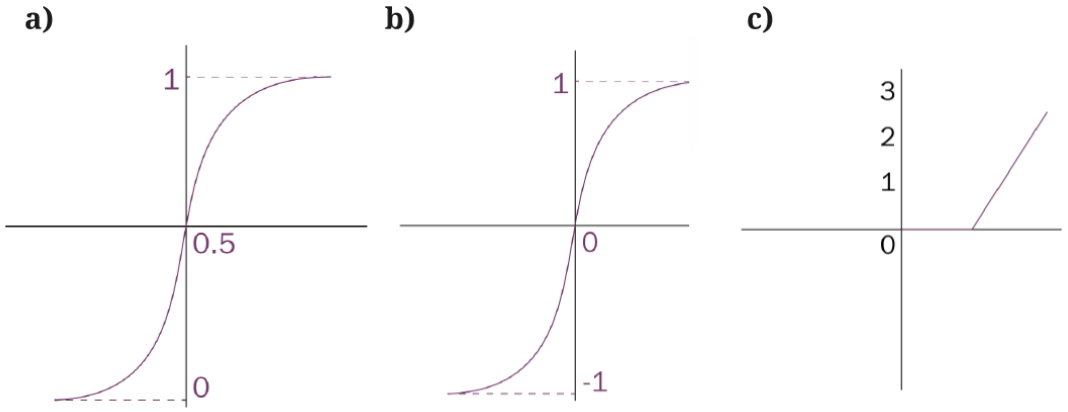

Figura 3.4: Funciones típicas de activación: Sigmoidal (a), Tangente Hiperbólica (b) y Función Lineal Rectificadora Unitaria (c) [30].

3.3.2 Entrenamiento de la red neuronal

El rendimiento de la red depende esencialmente de qué tan bien se adecúen los valores de los pesos W y los sesgos B. Dadas las dimensiones típicas de estas matrices (generalmente demasiado grandes como para determinar estos valores de forma analítica), se suele recurrir a lo que se conoce como retro-propagación, un algoritmo que iterativamente actualiza los pesos de cada capa.

Para comenzar, tanto los sesgos como los pesos se inicializan aleatoriamente. Se provee a la red con los datos de entrada y estos atraviesan todas las capas hasta la capa de salida. Este proceso se conoce como propagación hacia adelante o simplemente propagación. Al terminar esta instancia, se evalúan los valores devueltos por la red ($\hat{y}$) a través de lo que se conoce como función perdida o función costo. Esta función sirve para evaluar qué tan bien está funcionando la red. Una función costo utilizada comúnmente es el error cuadrático medio que suele abreviarse como MSE por sus siglas en inglés (*mean square error*). Se define como el promedio de la diferencia entre el valor objetivo (y) y el valor que predice la red ($\hat{y}$):

$$L = \frac{1}{2}(y - \hat{y})^2 \tag{3.16}$$

El objetivo de la retro-propagación es encontrar pesos que minimicen la función costo para que la red pueda predecir de forma apropiada (o al menos cada vez más apropiada) los valores $\hat{y}$. Entonces, una vez que se llega a la capa de salida se evalúa la función perdida para actualizar los pesos de acuerdo al resultado obtenido. Es en este punto donde entra en juego el método de descenso por gradiente mencionado anteriormente. En la figura 3.5 se presenta esquemáticamente la función costo en función de los pesos. Utilizando descenso por el gradiente, los pesos se modifican de acuerdo a la regla:

$$w = w - \alpha \frac{\partial L}{\partial w} \tag{3.17}$$

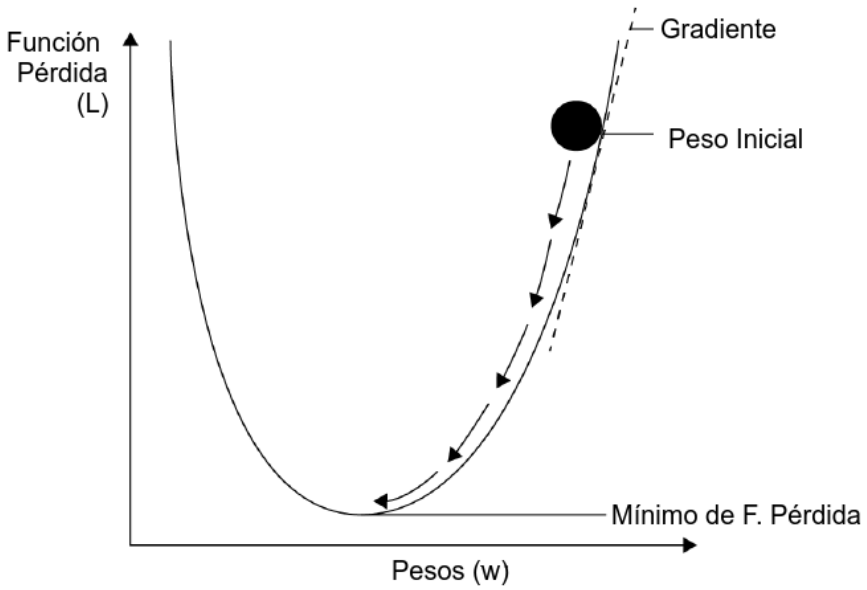

Figura 3.5: Esquema simplificado del descenso por gradiente: El peso inicial (círculo negro) se mueve a lo largo de la dirección en la que decrece el gradiente (línea punteada). De esta forma, se acerca al valor mínimo de la función pérdida. Imagen adaptada de la referencia [30].

con el objetivo de encontrar aquellos que minimicen L. Una explicación matemáticamente más rigurosa de esta minimización junto con una descripción más completa del mecanismo puede encontrarse en [30].

Como se describió en el capítulo anterior, al actualizar los parámetros del modelo en las direcciones por las que el gradiente de la función costo es grande y negativo, se hace tender a la función hacia un mínimo. Se denomina retro-propagación a esta actualización de los pesos de la red a través de la minimización de la función pérdida.

La mayoría de las librerías de aprendizaje por refuerzos utilizan modificaciones del método de descenso por gradiente para lograr que la minimización sea más eficiente. Por ejemplo, la librería Keras (una de las más típicas y la que se usa en este trabajo), implementa un algoritmo denominado ADAM. Este, utiliza un descenso por gradiente estocástico basado en lo que se conoce como "momentos" que guardan información sobre la dirección en la que se va moviendo el algoritmo [32].

3.3.3 Deep Q-learning

Dentro del Deep RL, uno de los algoritmos más utilizados es el *Deep Q-Learning* o DQL. Como se mencionó previamente, en un algoritmo de aprendizaje por refuerzos la relación Q de la ecuación (3.4), que da la correspondencia entre cada par estado-acción y su valor asociado, suele cargarse en una tabla. Al tener almacenados los valores asociados a cada acción para un estado, en cada paso se elige la acción que de el máximo valor para el mismo.

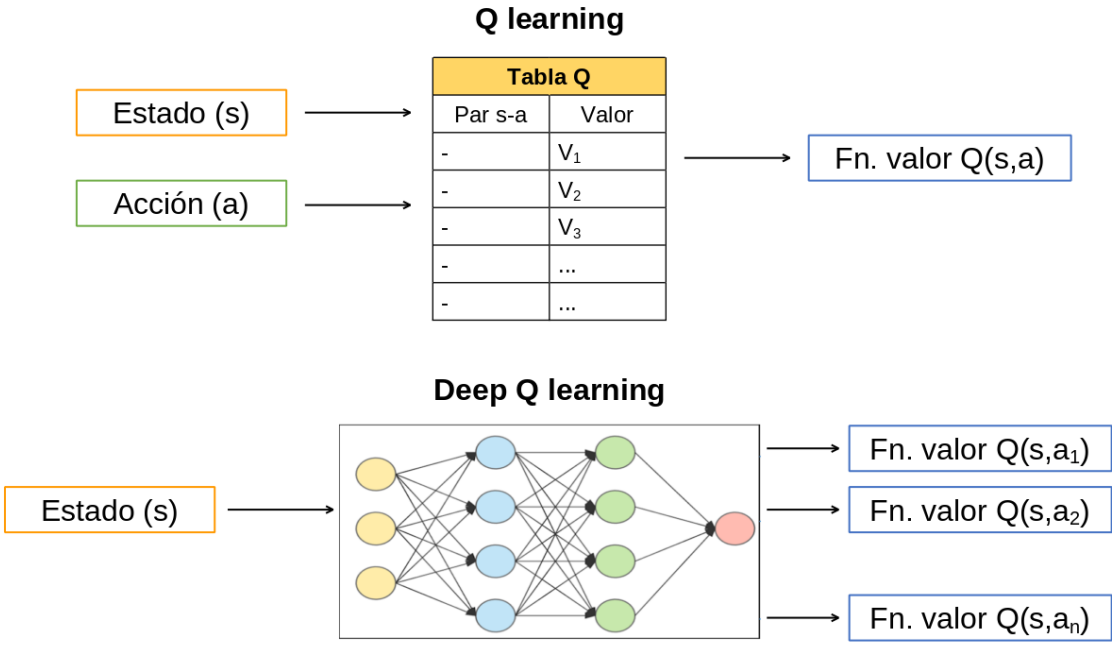

Figura 3.6: Comparación esquemática de los modelos de Q-learning (superior) y Deep Q-Learning (inferior). Dado un estado y una acción, los algoritmos de Q-Learning obtienen la función Q a través de una tabla. Por otro lado, en el DQL, una red neuronal toma un estado y obtiene la función Q para cada acción posible.

En el DQL, se reemplaza esta tabla con una red neuronal. Esto resulta especialmente conveniente en problemas de grandes dimensiones donde la construcción de una tabla de este tipo resultaría demasiado compleja. Utilizando una red puede aproximarse el valor de cada una de las acciones a través de los pesos (W).

En este caso, el entrenamiento de la red significa resolver la ecuación de Bellman para Q dada por (3.8) minimizando la función de perdida construida a partir de la diferencia con la función de valor Q óptima.

Para explicitar este procedimiento utilizando expresiones simples, se describe esta minimización para sistemas deterministas, es decir, aquellos en los que dado un estado s bajo el efecto una acción a evolucionan indefectiblemente a un estado s'[3]. En estos casos, la ecuación de Bellman para Q puede escribirse como:

$$Q_*(s,a) = r(s,a) + \gamma \max_a Q_*(s',a',W) \tag{3.18}$$

Se define entonces la función pérdida como:

$$L = [\hat{y}_i - Q(s,a;W)]^2 \tag{3.19}$$

donde Q representa el valor que predice la red (lo que en la sección previa denotamos como $\hat{y}$) y y_i representa en cada paso el valor "objetivo".

[3]En particular, el problema de transmisión de estados cuánticos que se trata en este trabajo entra en esta categoría.

En el DQL, el agente realiza dos tareas: la ejecución y el aprendizaje. En primer lugar, se inicializan de forma aleatoria los pesos W de la red de evaluación. Los pesos $\hat{W}$ de la red objetivo se inicializan como $\hat{W} = W$.

Luego, para cada episodio de ejecución, el agente elige una acción a_t que lleva a un estado s_{t+1}. Ahora, en función de este estado se elegirá la siguiente acción. De acuerdo con este nuevo estado, se genera una recompensa r_t. Esta "experiencia" se almacena en la memoria del agente, generalmente como un vector $e_t = (s_t, a_t, r_t, s_{t+1})$. Estos episodios se repiten hasta que se alcance alguna condición de tolerancia (o un máximo de iteraciones). Llegado ese momento, el sistema se reinicia y comienza un nuevo episodio de ejecución. Hasta aquí la descripción del algoritmo es idéntica a la del aprendizaje por refuerzos clásico, la diferencia es que cada determinada cantidad de episodios de ejecución, ocurre una instancia de aprendizaje. En estas instancias, el agente recupera un conjunto de experiencias de la memoria y a través de retropropagación actualiza los pesos de la red de evaluación W. Esto le permite al agente encontrar una política óptima (resolver las ecuaciones de Bellman) de manera más eficiente, en especial en sistemas de muchas dimensiones donde construir una tabla Q sería muy costoso computacionalmente.

Parte II

Implementación y Resultados

Capítulo 4

Configuración de Acoplamientos en Modelos Independientes del Tiempo

En este capítulo, se analiza la transmisión en dos modelos independientes del tiempo para cadenas de espines 1/2: el Hamiltoniano XX (ec. 1.13) y el de Heisenberg (ec. 1.22). En estos sistemas, la estrategia utilizada para maximizar la fidelidad es la de obtener conjuntos óptimos de valores para los coeficientes de intercambio o acoplamientos J_i, es decir, se busca un vector $\mathbf{J}^N$ que maximice la probabilidad de transición o fidelidad (F_N, ec. 1.10). El método que se propone para ello es la implementación del algoritmo genético, descripto extensivamente en la sección 2.3 del presente trabajo[1]

En primer lugar, se analiza el Hamiltoniano XX. Previamente, se mencionó que este sistema permite transmisión perfecta [17] a través de la configuración individual de los acoplamientos. Por eso, resulta un marco de referencia útil para verificar la validez de los métodos de optimización. En segundo lugar, se extiende la implementación de este algoritmo al Hamiltoniano de Heisenberg, donde se analiza el aumento en la complejidad del sistema respecto al modelo anterior.

Al estar trabajando en sistemas independientes del tiempo, se mantiene fijo el tiempo de llegada y se optimiza únicamente sobre los acoplamientos. Es decir, al optimizar la probabilidad de transición $F_N(t, \mathbf{J}^N)$, el valor de t se mantiene constante. En el caso del Hamiltoniano XX, dado que se sabe que puede obtenerse transmisión perfecta para cualquier tiempo final [17] se toma $t = 1$. Por otro lado, tomando como referencia los resultados existentes ([21], [11]) con los que se ha obtenido PGST en el modelo de Heisenberg se considera $t = N$.

[1]Los códigos implementados para obtener los resultados de la presente sección pueden encontrarse en: www.github.com/sofips/Genetic-Algorithm

4.1 Implementación general del Algoritmo Genético.

Para implementar el algoritmo genético, se utilizó la librería open-source de Python: PyGAD [26]. En los sistemas estudiados, los individuos están dados por los vectores $\mathbf{J^N}$ con los acoplamientos del sistema. A su vez, cada acoplamiento representa un gen. Dado que las cadenas estudiadas son centro-simétricas, se optimiza sobre la mitad de los acoplamientos totales tomando cadenas con largos pares.

Para ejecutar el algoritmo, se crea un objeto o instancia de la clase `GA`, a la que deben ingresarse los siguientes parámetros:

- `num_generations` = Número de generaciones sobre las que itera el algoritmo, es decir, la cantidad de veces que se genera una nueva población.
- `num_parents_mating` = Número de individuos "padres" que se seleccionan para dar origen a la siguiente generación
- `fitness_func` = Función aptitud o *fitness*, en este caso, naturalmente, se basa en la probabilidad de transición. En las siguientes secciones, se discuten las diferencias entre utilizar una función exactamente igual a dicha probabilidad o añadir a la misma factores asociados a características físicas del sistema.
- `sol_per_pop` = Cantidad total de soluciones o individuos en la población, es decir, el número total de vectores $\mathbf{J^N}$ en los que se calcula la aptitud en cada generación.
- `num_genes` = Cantidad de genes en un individuo, en este caso, será igual la mitad del largo del vector $\mathbf{J^N}$, es decir, el número de acoplamientos a optimizar.
- `init_range_low/high` = Parámetros de inicialización de la población. Los genes se inicializan con valores aleatorios dentro de estos límites. Es posible, sustituir estos dos parámetros por el atributo `initial_population` para inicializar la población con un conjunto de vectores personalizado. Además, puede configurarse el atributo `gene_space` que limita los valores que pueden tomar los genes a lo largo de su evolución.
- `parent_selection_type` = Método de selección de padres.
- `keep_parents` = Indica cuántos de los padres mantener en la siguiente generación. Para mantener todos puede igualarse a -1, para eliminar todos a 0 y cualquier número positivo significa mantener esa cantidad.
- `crossover_type` = Tipo de cruza o *crossover*.
- `mutation_probability` = Probabilidad de que ocurra *crossover*.
- `mutation_type` = Tipo de mutación.
- `mutation_probability` = Probabilidad de que ocurran mutaciones.

Una vez configurados cada uno de los parámetros y condiciones, un método de la clase denominado `run` permite ejecutar la instancia generada. Luego, quedan almacenados en la misma instancia los resultados de interés: el valor máximo de *fitness* alcanzado, la solución que se corresponde con dicho valor y el número de generaciones realizados, entre otros. La librería utilizada cuenta con muchas otras funcionalidades y herramientas de visualización que pueden consultarse en [26].

4.2 Hamiltoniano XX

Para ilustrar el funcionamiento de la librería PyGAD, se describe la evolución de la fidelidad junto con los acoplamientos obtenidos para una cadena de 24 espines. En la figura 4.1, se muestra cómo aumenta la fidelidad a medida que se realizan iteraciones del algoritmo genético. Este experimento se realizó con las funciones por defecto de la librería, tomando una población de 500 individuos (vectores de acoplamientos) inicializados de forma aleatoria. Se realiza una selección de 50 individuos como padres, a través del método 'sss'. Luego, con una probabilidad de 0.5, se genera *crossover* punto a punto. También, con la misma probabilidad, ocurren mutaciones aleatorias que suman a los genes valores entre -1 y 1. Se fija la tolerancia en 10^{-3}. En este caso, la fidelidad se utiliza a su vez como función *fitness*. Los puntos en el gráfico en 4.1 representan la probabilidad de transmisión más alta presente en la población para esa generación.

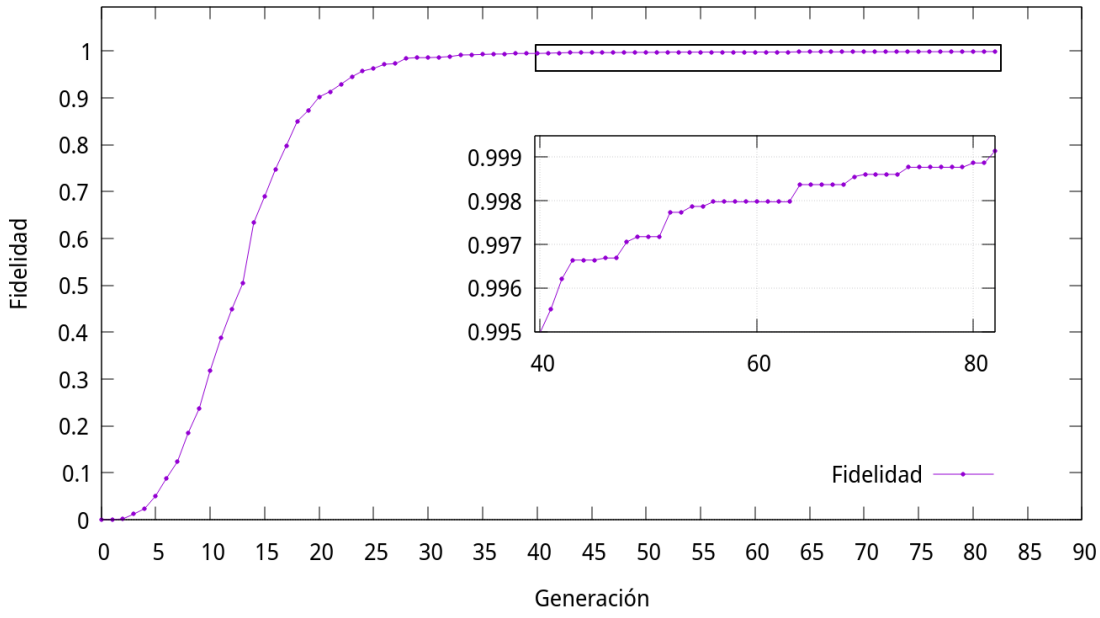

Figura 4.1: Evolución de la fidelidad asociada a la transmisión para una cadena de 24 espines con interacciones modeladas por el Hamiltoniano XX. Se realiza un acercamiento a la porción final de la curva para destacar que el valor continúa aumentado hasta alcanzar la tolerancia fijada.

Inicialmente, se puede ver que ningún individuo permite obtener una transmisión de

alta fidelidad. En menos de 25 generaciones, se observa que, para al menos un individuo, la probabilidad de transición supera el 90 %. Finalmente, el algoritmo alcanza la tolerancia establecida (99.9 %) en 82 generaciones.

A continuación, en la figura 4.2, se muestra el conjunto de acoplamientos obtenido como resultado. Puede notarse que los acoplamientos obtenidos se acercan pero no coinciden con los valores de la solución analítica[2] de transmisión perfecta dada por (1.20). De todas formas, es esperable que la solución obtenida sea distinta ya que el conjunto de acoplamientos que permiten PST (o PGST) no es necesariamente único.

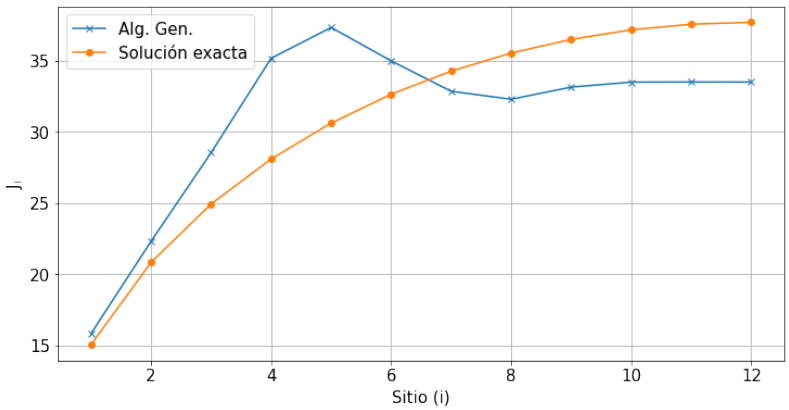

Figura 4.2: Valores de los acoplamientos encontrados (azul) asociados a la máxima fidelidad obtenida (99.9 %) utilizando el algoritmo genético. Se comparan con los valores del modelo de transmisión perfecta presentado por [17]. La forma analítica de esta solución es $J_i = \pi\sqrt{i * (N - i)}$, donde i es el número de sitio correspondiente.

4.2.1 Influencia del espacio de inicialización.

En la figura 4.3, se muestra la fidelidad máxima alcanzada para cadenas donde los genes fueron inicializados con valores que van de 1 a N, 2N y 3N, siendo N el largo de la cadena. Se mantuvieron los parámetros y funciones a excepción de la probabilidad de mutación, que se llevó a 0.8, y el tamaño de la población, que se disminuyó a 200. Se observa que para cadenas cortas es posible optimizar los acoplamientos asociados a la cadena para cualquier inicialización. Al agregar espines a la cadena, es necesario utilizar espacios de inicialización mayores. Por ejemplo, para cadenas con más de 28 espines, no se obtienen resultados favorables con el menor espacio de inicialización [1,N]. Al ampliarlo hasta 2N, la optimización funciona para largos de cadena menores a 50. Finalmente, se muestra que para cadenas de hasta 70 espines fue suficiente inicializar los genes entre 1 y 3N. Con dichos parámetros, es posible obtener un conjunto de acoplamientos que resulta en una fidelidad del 99 %.

No se encontró un motivo para este comportamiento vinculado estrictamente a la física del problema. Sin embargo, es razonable pensar que al aumentar la cantidad de variables de la función a optimizar sea necesario aumentar el espacio de búsqueda para encontrar

[2]Notar que la fórmula analítica de los valores J_i presente en la referencia [17] difiere en un factor 1/2 de la ecuación (4.1), esto se debe a las definiciones de las constantes de acoplamiento y los Hamiltonianos

un conjunto adecuado de valores. Se realizaron experimentos para estudiar los efectos de variar otros parámetros y funciones de la librería junto con el largo de la cadena. Sin embargo, se observó que este efecto sobre la fidelidad obtenida no cambia al modificar los métodos de cruza, selección de padres o mutación. También se implementaron variaciones de la función *fitness* y se concluyó que, en este modelo, no influyen sobre la fidelidad final alcanzada. Sin embargo, en la siguiente sección se estudia cómo influyen en la suavidad de las soluciones obtenidas.

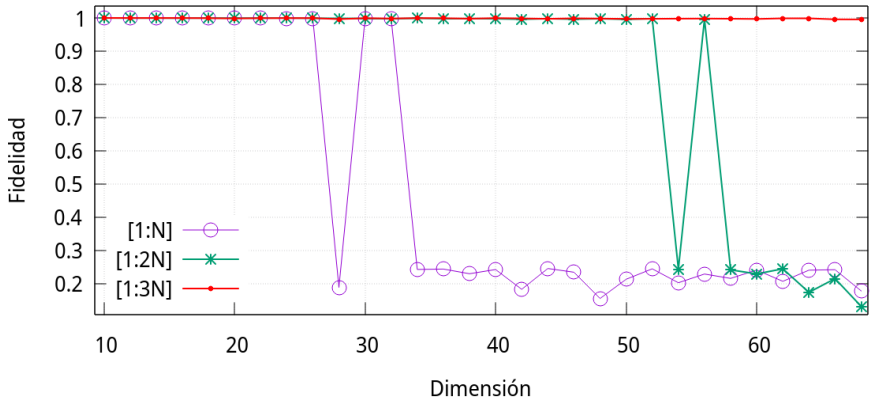

Figura 4.3: Fidelidad alcanzada para cadenas de espines 1/2 de distintos largos con interacciones dadas por un Hamiltoniano XX partiendo de inicializaciones diferentes.

Tiempo de ejecución

En la figura 4.4, se muestra el tiempo de cómputo necesario para implementar el algoritmo. En particular, se muestra el caso de inicialización entre 1 y 3N. Se observa que en todos los casos el tiempo de cómputo es menor a 1500 s. (25 minutos). Para dar idea del orden de crecimiento de los tiempos, se añade a la figura una curva proporcional a N^2. Sin embargo, el tiempo de cómputo no sigue una ley determinista debido a que es un método estocástico.

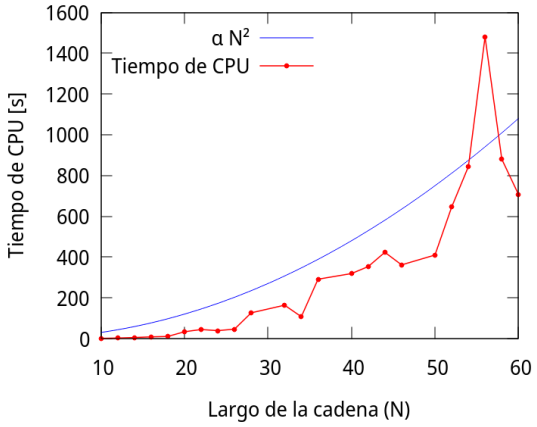

Figura 4.4: Tiempo de CPU para la optimización en el modelo XX.

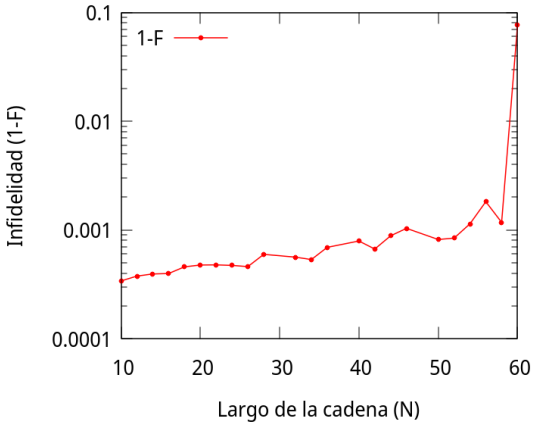

Figura 4.5: Infidelidad correspondiente a los tiempos de la figura 4.4.

Además de depender del largo de la cadena, el tiempo que tarda en ejecutarse el algoritmo depende de cuántas generaciones le toma alcanzar la tolerancia preestablecida (si es que lo hace). Para obtener estos gráficos, se fijó la tolerancia en 0.001[3]. Se puede ver, en la figura 4.5, que el algoritmo no alcanza este valor para dimensiones mayores a 30. Sin embargo, permite obtener una probabilidad de transmisión mayor al 99 % incluso para las cadenas más largas. Al no alcanzar la tolerancia, las iteraciones se terminan si se ejecuta el número máximo de generaciones o si se produce "saturación", es decir, si para un determinado número de generaciones el valor de aptitud máximo no cambia. Estos resultados se obtuvieron tomando como saturación 100 generaciones sin cambios y estableciendo un total de 1000 generaciones como máximo. Se promedia sobre 10 corridas del algoritmo para obtener cada punto.

4.2.2 Efecto de las variaciones en la función *fitness*.

Como se mencionó anteriormente, es posible añadir a la función *fitness* factores vinculados a las propiedades físicas del problema. En esta sección se proponen dos funciones: una asociada a la suavidad de la solución y otra asociada al ordenamiento de los espectros de energías.

En la ecuación (4.1), se define una función *fitness* que añade un factor para considerar la estabilidad en los acoplamientos:

$$A_{\bar{J}} = \ F \exp\left(-\sum_{i=2}\left(\frac{J_i - \overline{J}}{J_{max}}\right)^2\right) \tag{4.1}$$

Esta condición se inspira en la estrategia previamente descripta de utilizar cadenas homogéneas en el centro y con acoplamientos optimizados en los bordes. Buscando reproducir este comportamiento, se dejan "libres" las interacciones de los dos primeros espines de la cadena y se añade a los centrales la condición de estar cercanos entre sí. Para maximizar esta función, debe maximizarse tanto la probabilidad de transmisión F como el factor exponencial cuyo valor es inversamente proporcional a las diferencias de cada uno de los acoplamientos al valor promedio de todos ellos.

Por otro lado, se define una función aptitud basada en el espectro:

$$A_{E_n} = \ F \exp\left(\sum_{i=1}\left(\frac{\Delta E_i - int(\Delta E_i)}{\max(\Delta E_i)}\right)^2\right) \tag{4.2}$$

En este caso, se busca incorporar que las diferencias entre energías sucesivas, $\Delta E_i = E_{i+1} - E_i$, sean cercanas a números enteros. De esta forma, se tiene en cuenta el ordenamiento en los espectros que caracteriza a los sistemas con transmisión perfecta discutido en la introducción.

En la figura 4.6, se muestran los valores finales de acoplamientos obtenidos utilizando las distintas opciones de funciones *fitness* e inicializando en el intervalo [1,3N] para una

[3]El resto de los parámetros coinciden con los utilizados para obtener la figura 4.1

cadena de N=24. Se compara también con la solución obtenida utilizando una *fitness* que coincide con la fidelidad pero inicializando en el intervalo [1,N].

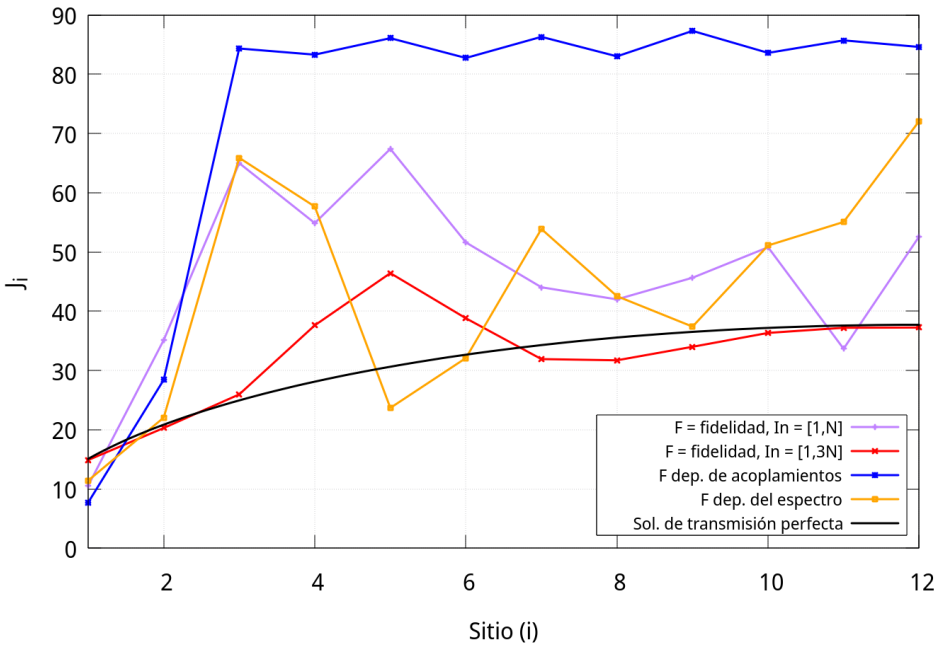

Figura 4.6: Acoplamientos obtenidos para distintas funciones *fitness* en una cadena de N=24. Se comparan los valores de acoplamientos para la función *fitness* que se corresponde con la fidelidad tomando diferentes inicializaciones y los obtenidos para las funciones *fitness* dadas por (4.1) y (4.2).

A pesar de tomar valores diferentes, cada uno de los conjuntos de acoplamientos permite alcanzar una fidelidad de más del 99.9 %. Se observa que la solución que más se acerca a la del trabajo de referencia [17] es la que inicializa en el menor intervalo. Además, se puede ver que la solución que utiliza una función *fitness* basada en la estabilidad o suavidad de la solución, es decir, aquella dada por la ecuación (4.1), efectivamente permite encontrar un conjunto que incorpora dicho comportamiento.

En la figura 4.7, se muestran los autovalores correspondientes a construir el Hamiltoniano XX a partir de cada una de las soluciones de la figura 4.6. . Dentro de las soluciones obtenidas, se observa que la que más respeta la condición de un espectro ordenado y equiespaciado es la correspondiente a inicializar en el intervalo más chico. Esto es acorde también a los valores de los acoplamientos que son los que más se acercan a los de la solución analítica para la transmisión perfecta. Si bien aplicando la *fitness* dependiente de la energía no se logra un espectro tan ordenado, sí se observa este comportamiento en la región central del espectro.

Como menciona el trabajo [21], es crucial que la porción central esté ordenada para lograr una alta fidelidad. El gráfico en 4.7 corrobora la hipótesis de que para obtener transmisión casi perfecta (PGST) es suficiente con que haya porciones del espectro que cumplan esta condición. Esto cobra importancia especialmente para el caso del Hamiltoniano de Heisenberg que se estudia en la siguiente sección, donde un cálculo analítico del

espectro como el realizado en [17] resulta mucho más complejo [33].

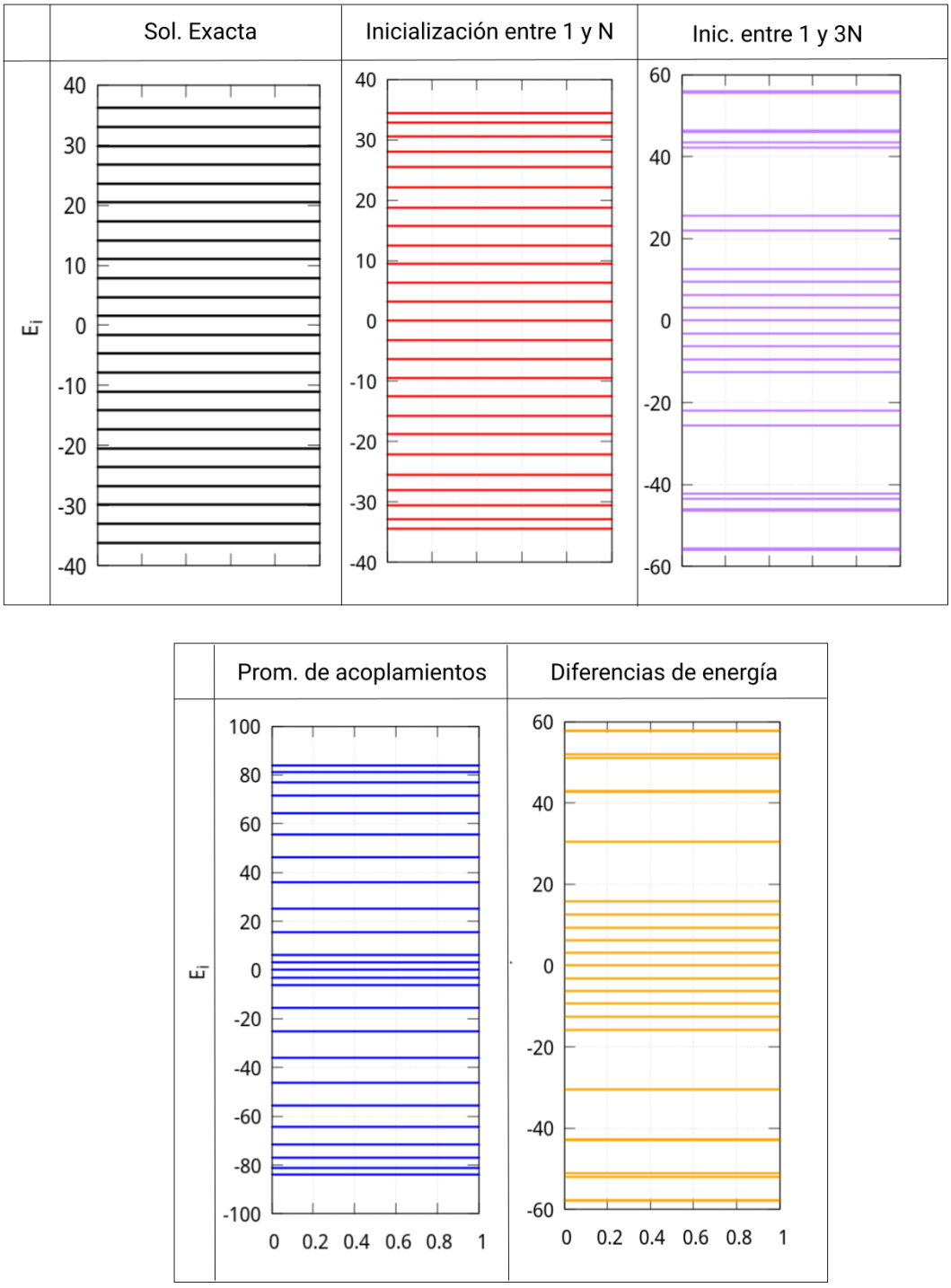

Figura 4.7: Espectros obtenidos para las distintas funciones *fitness* para una cadena de N=24. Se corresponden a los autovalores de los Hamiltonianos XX construidos a partir de los acoplamientos en la figura 4.6.

4.3 Hamiltoniano de Heisenberg.

A continuación, se implementó el algoritmo genético para el Hamiltoniano de Heisenberg dado por (1.22). Para estudiar el aumento de complejidad al pasar de un sistema al otro, se estudian los resultados que se obtienen a partir del hamiltoniano general en (1.11) con $\gamma = 0$ e incrementando el valor de Δ gradualmente entre 0 (Hamiltoniano XX) y 1 (Hamiltoniano de Heisenberg). Se calcula la fidelidad máxima obtenida con las funciones por defecto y una *fitness* igual a la fidelidad para distintos largos de cadena y diferentes valores de Δ. Se mantienen los parámetros por defecto, tomando probabilidades de mutación y *crossover* iguales a 0.5.

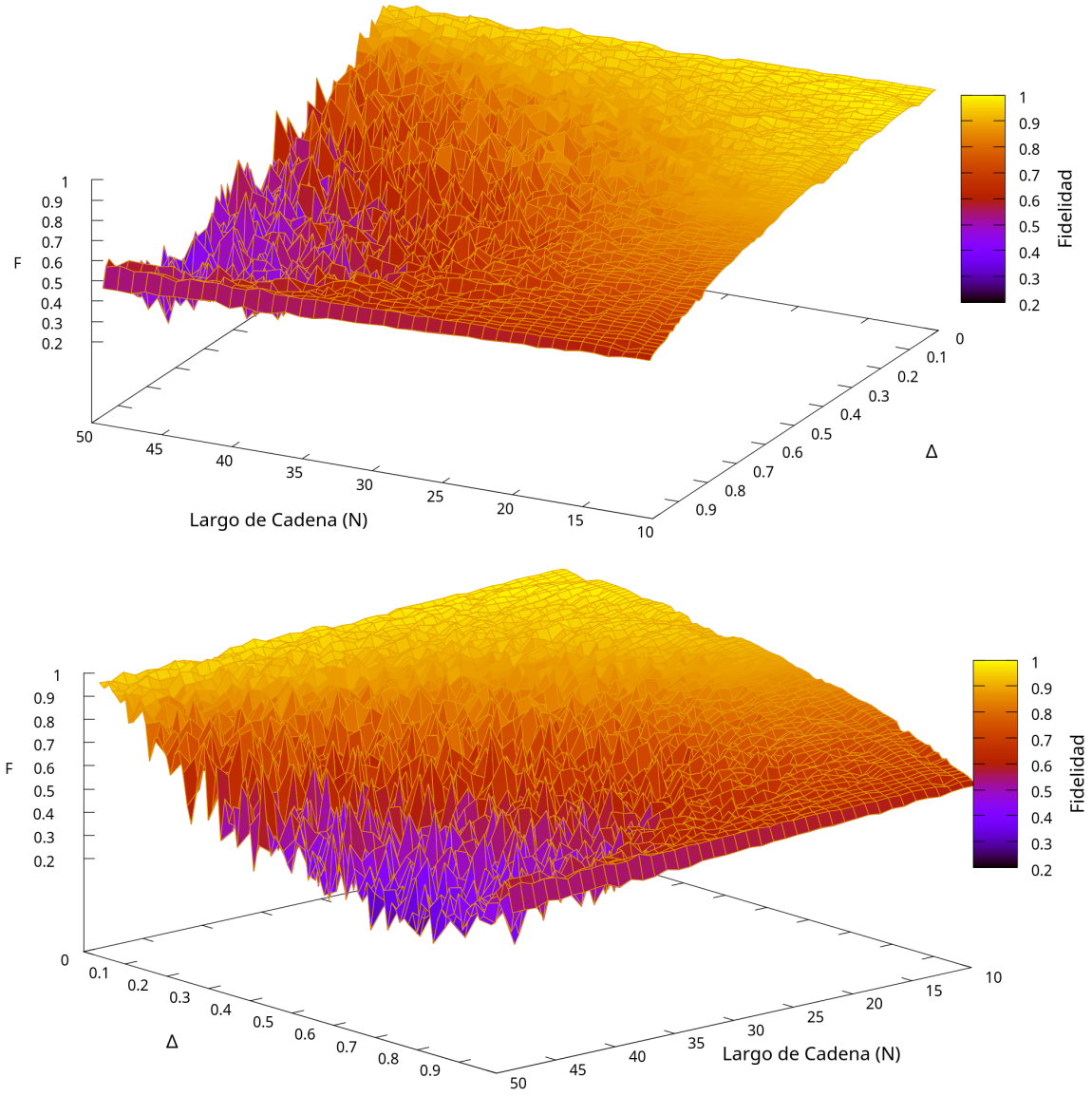

Figura 4.8: Fidelidad máxima obtenida para distintos largos de cadena y distintos valores del parámetro Δ. El gráfico inferior representa una rotación en $-90°$ del superior para una mejor visualización.

La figura 4.8 ilustra cómo disminuye la fidelidad máxima tanto al aumentar el largo de la cadena como al aumentar el valor de Δ. Como se esperaba, dada la forma "desordenada" en que se transmite la información a lo largo de una cadena de Heisenberg, se muestra que obtener una buena transmisión para $\Delta \sim 1$ resulta más complejo que en el modelo XX. Además, se ilustra la rugosidad del espacio de optimización ya que en ambos cortes se puede observar la aparición de picos pronunciados, especialmente para los mayores largos de cadena.

Con el objetivo de mejorar las soluciones obtenidas, se implementó, en primer lugar, la función *fitness* dependiente de las diferencias en el espectro de energía dada en (4.2). También se estudió la posibilidad de incorporar funciones mutación personalizadas. En particular, se utilizó una función que elegía un acoplamiento al azar y redefinía su valor según:

$$J_i = \begin{cases} i = 1, \qquad = J_1 * (1 + \alpha), \alpha \in (-0.1, 0.1) \\ \\ i \in [\![2, N]\!] = J_i * (1 + \beta), \beta \in (-0.5, 0.5) \end{cases} \tag{4.3}$$

donde α y β son valores aleatorios en los intervalos dados. Esta mutación personalizada permite que las variaciones a los valores sean acordes al valor del acoplamiento afectado en lugar de utilizar un valor fijo como la mutación *random* predeterminada por PyGAD. Se diferencia el acoplamiento inicial ya que algunos de los trabajos previos [3] han mostrado que es conveniente que el extremo tenga una interacción débil con el resto de la cadena.

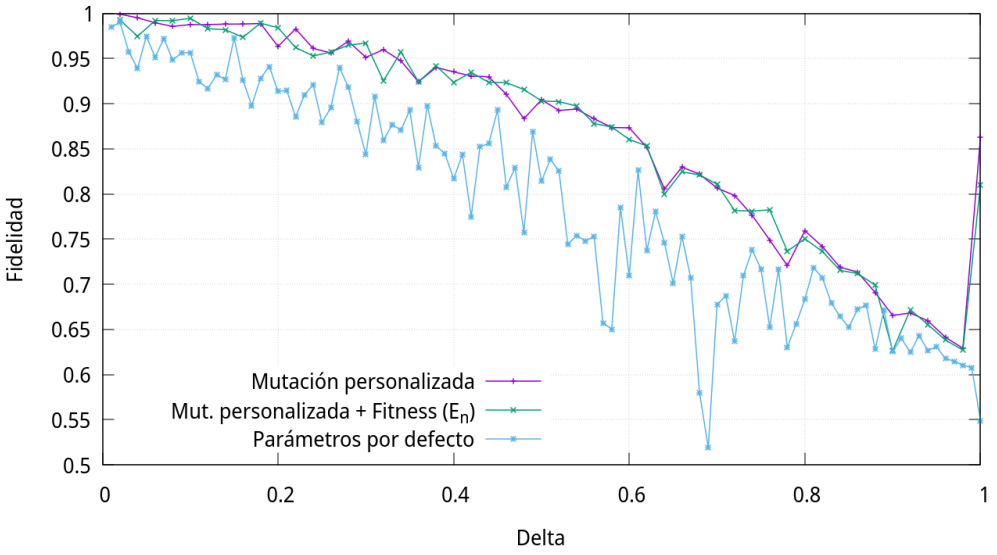

Figura 4.9: Fidelidad obtenida aplicando diferentes variaciones del algoritmo genético para optimizar cadenas de 24 espines para distintos valores del parámetro Δ. Se compara la implementación de la función *fitness* igual a la fidelidad (celeste) y la dada por (4.2) con (verde) y sin mutación personalizada (violeta).

En la figura 4.9, se compara el valor máximo de fidelidad obtenido para distintos valores de Δ en una cadena de 24 espines. Efectivamente, la incorporación de estas modificaciones

permite suavizar los resultados obtenidos y mejorar la fidelidad máxima lograda. No se observan diferencias significativas al reemplazar la función fidelidad por (4.2) como *fitness*, de modo que se infiere que es más significativo el efecto de la mutación personalizada.

Finalmente, se estudia la dependencia de los resultados y el tiempo de cómputo con el largo de las cadenas. En la figura 4.11, se muestra el valor de infidelidad alcanzado junto con los tiempos de CPU necesarios (fig. 4.10) para distintos tamaños. Se observa que al modificar las funciones mutación y *fitness*, es posible mantener un buen valor de fidelidad independientemente de las dimensiones. Se puede ver que la implementación de funciones personalizadas provoca un aumento en el tiempo de cómputo. Sin embargo, al no superar las dos horas, continúa siendo un método eficiente, incluso al incorporar estas modificaciones que, además, mejoran los resultados.

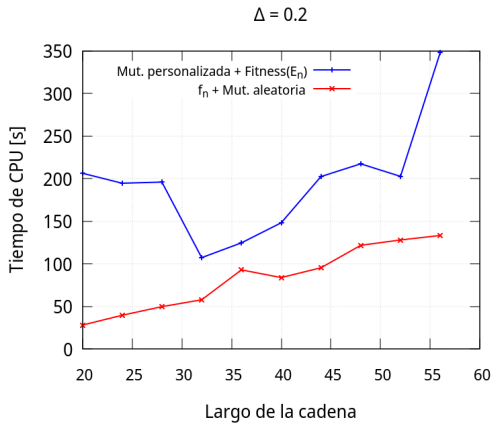

Figura 4.10: Tiempo de CPU para implementar diferentes variantes del algoritmo para el Hamiltoniano de Heisenberg. Se compara la implementación de las funciones por defecto (mutación aleatoria y fitness exactamente igual a la fidelidad, curva roja) con las funciones personalizadas (mutación customizada y fitness dependiente del orden del espectro, curva azul).

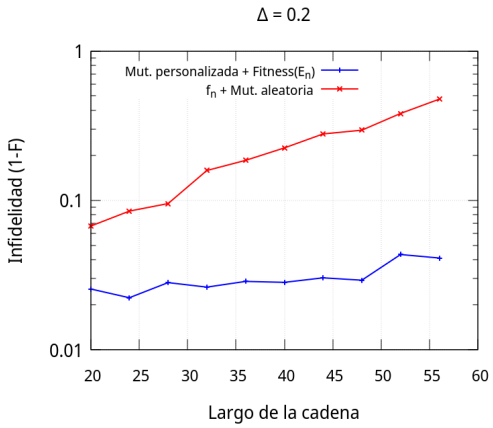

Figura 4.11: Infidelidad correspondiente a los tiempos de la figura 4.10.

Capítulo 5

Control Dinámico utilizando Deep Reinforcement Learning

En este capítulo, se busca implementar el algoritmo desarrollado en el trabajo de Zhang et al. [19] que explora una nueva estrategia para la transmisión de estados basada en el Hamiltoniano dependiente del tiempo dado por (1.24). En este modelo, los acoplamientos se mantienen constantes y la optimización se realiza sobre los valores de campo magnético externo. Es decir, se busca una secuencia óptima de pulsos magnéticos que logre que la transmisión se realice en el menor tiempo y con la mayor fidelidad posible. La técnica de optimización que se propone para ello es el Deep Reinforcement learning o *DRL* descripto en el capítulo 3. En particular, se utiliza un algoritmo de Deep Q-learning con el objetivo de que el agente logre aprender una política que le permita elegir una secuencia de campos magnéticos que provoque una transmisión de alta fidelidad.

5.1 *DRL* aplicado al problema de transmisión de estados.

5.1.1 Elementos principales del algoritmo

Para resolver un problema físico con aprendizaje por refuerzos, en primer lugar, es necesario definir la correspondencia entre los elementos propios de este tipo de algoritmos y las partes del sistema físico en cuestión. A continuación, se definen los espacios de estados y acciones junto con el modelo asociado.

Espacio de estados

Naturalmente, el estado del sistema está dado por la función de onda asociada a la cadena de espines. Se trabajan estos estados como vectores en la base de una excitación, donde el estado inicial del sistema está dado por el vector $|1\rangle = (1, 0, 0, ...)$ de la misma. A medida que evoluciona, el estado estará dado por los coeficientes correspondientes a la excitación en cada sitio.

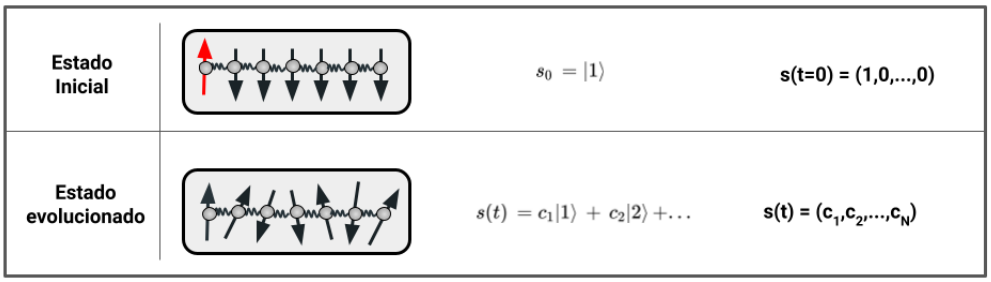

Figura 5.1: Representación del estado inicial y su evolución en la base de una excitación.

Espacio de acciones

Como espacio de acciones se toma el utilizado en la referencia [19], allí se definen 16 acciones según:

$$\begin{cases} a_t = \frac{1}{B}[2^0 B_1(t) + 2^1 B_2(t) + 2^2 B_3(t)] \quad \text{para } 0 \leq a_t < 8 \\ \\ a_t - 7 = \frac{1}{B}[2^0 B_{N-2}(t) + 2^1 B_{N-1}(t) 2^2 B_N(t)], \quad \text{para } 8 \leq a_t < 15 \end{cases} \tag{5.1}$$

donde $B_i(t)$ representa el campo aplicado sobre el i-ésimo elemento de la cadena y puede tomar únicamente los valores $0, B$. Además, se toma $B_i = 0$ para $3 \leq i \leq N-2$. De esta forma, las posibilidades consisten en encender uno o más de los campos correspondientes a los primeros o últimos 3 elementos de la cadena. En la figura 5.2, se esquematiza cada una de las acciones para mayor claridad. Las mismas se representan con una matriz calculada a partir de (1.24). La forma explícita de las 16 matrices obtenidas se presenta en el apéndice A.

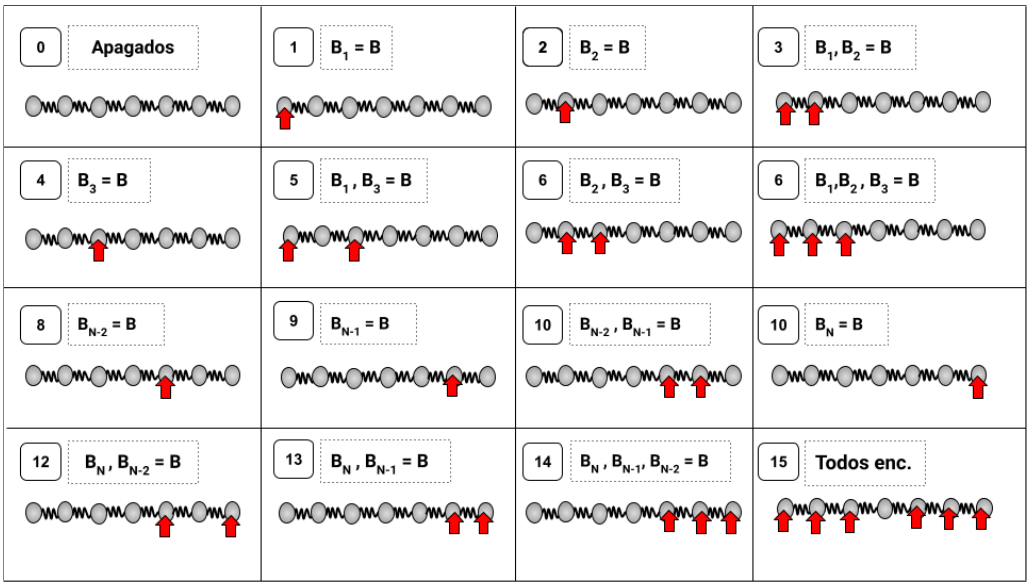

Figura 5.2: Esquema de las 16 acciones posibles definidas según (5.1). Las flechas rojas representan que un campo externo está siendo aplicado sobre ese elemento de la cadena.

Modelo y evolución temporal

El tiempo total τ se discretiza en N_τ pasos de longitud dt denotados $t_i, i = (0, N_\tau)$ y no está fijo como en el caso anterior. La evolución de cada estado al siguiente dependerá de la acción tomada en el i-ésimo intervalo de tiempo.

Para encontrar el estado evolucionado, se definen los propagadores asociados a cada una de las matrices de acción. Durante cada intervalo de tiempo dt, el Hamiltoniano se mantiene constante. Entonces, es posible construir el propagador a partir de los autovectores de cada matriz de acción como si fuera un sistema estacionario. De esta forma, si el sistema se encuentra en el estado $|\psi_i\rangle$ en el tiempo t_i y se toma la acción a definida por la matriz M_a, se construye el propagador según:

$$U_a(t) \;=\; \sum_j \left|E_j^a\right\rangle \; \left\langle E_j^a\right| \; e^{-i \; E_j^a (t - t_i)} \tag{5.2}$$

donde $\left|E_j^a\right\rangle$ es el j-ésimo autovector de la matriz asociada a la acción a. Considerando que el valor de $t_{i+1} - t_i$ es dt (constante), los propagadores toman la forma:

$$U_a \;=\; \left(\sum_j \left|E_j^a\right\rangle \; \left\langle E_j^a\right| \; e^{-i \; E_j^a dt} \right) \tag{5.3}$$

Luego, aplicar la acción y encontrar el estado resultante consistirá en multiplicar este propagador por el estado dado del sistema:

$$|\psi_{i+1}\rangle = U_a \, |\psi_i\rangle \tag{5.4}$$

Esta forma de evolucionar los estados resulta sumamente eficiente ya que pueden diagonalizarse las matrices y construirse los 16 propagadores asociados antes de comenzar el algoritmo de aprendizaje. De esta forma, quedarán almacenados y no será necesario recalcularlos para las distintas iteraciones o episodios.

Recompensa

Finalmente, se define la función de recompensa instantánea basada en la fidelidad. Se utiliza la misma función que en el trabajo de referencia, que está dada por:

$$r(t) = \begin{cases} 10 F_N(t), & F_N(t) \leq 0.8 \\ \frac{100}{1+\exp(10(1-\zeta-F(t)))}, & 0.8 \leq F_N(t) \leq 1-\zeta \\ 2500, & F_N(t) > 1-\zeta \end{cases} \tag{5.5}$$

donde ζ es la tolerancia establecida para $1 - F$.

5.2 Algoritmo utilizado

El algoritmo utilizado en este trabajo está inspirado en el aplicado por los autores del trabajo de referencia [19]. Los códigos utilizados en esta sección fueron programados en Python 3.10. Dado que el trabajo mencionado no proveía los mismos, gran parte de esta segunda etapa consistió en su desarrollo. Para ello, se utilizaron como referencia los códigos del repositorio en github de la referencia [34]. Los programas desarrollados para obtener los resultados de esta sección pueden encontrarse en: `www.github.com/sofips/QM_deep_rl`.

En primer lugar, es necesario inicializar la memoria R del sistema como un vector vacío y la red neuronal Q con pesos aleatorios. Luego, se inicializa el sistema con el estado de una excitación y comienzan los episodios del algoritmo. Cada episodio implica pasar del estado inicial al final, evolucionando a lo largo de N_τ pasos temporales. En cada uno de estos, el agente selecciona una acción de acuerdo a la política ϵ-*greedy* y la ejecuta. Al hacerlo, observa la recompensa y el estado resultante, y almacena esta "experiencia" en la memoria R. Cada l pasos temporales, ocurre una instancia de aprendizaje durante las cuales se recupera un "*minibatch*" de experiencias que se utiliza para actualizar la red. A continuación, se muestra un esquema general del algoritmo descripto:

Algoritmo DRL aplicado a la transmisión de estados cuánticos

Inicializar la memoria R
Inicializar la red Q con pesos aleatorios

Para cada **episodio** : ▷ **episodio** ≡ Evol. del s inicial al final

Inicializar estado del sistema

Para i = 0, N_τ : ▷ pasos temporales → $\tau = N_\tau$ dt
El agente selecciona una acción a con probabilidad ϵ
Se ejecuta la acción mediante $s_{i+1} = U_a \cdot s_i$
Se calcula la recompensa
Se almacena el vector de experiencia $E_i = (s_i,\ a_i,\ r, s_{i+1})$

Para cada inst. de aprendizaje : ▷ cada l pasos temporales
El agente recupera un *minibatch* de experiencias.
Se actualiza la red

5.2.1 Detalles de implementación

Para implementar el ambiente o *enviroment* con los elementos descriptos en la sección previa, se diseña una clase utilizando la librería `gym` de openAI [35]. Para comenzar, se crea una instancia de la misma, definiendo el espacio de estados, el de acciones, los propagadores asociados a las mismas, la función de recompensa y un método denominado `reset` que permite llevar al sistema al estado inicial. Dentro de la clase ambiente, también se especifican otros parámetros del sistema como la longitud y cantidad de pasos temporales,

la tolerancia ζ y la magnitud de los campos magnéticos. El largo de cadena es el único parámetro que se ingresa desde afuera de la clase.

También, se define la clase agente, con los métodos necesarios para elegir acciones en cada episodio y almacenar información en la memoria. Al crear una instancia de esta clase, se especifican los valores de los hiper-parámetros de aprendizaje:

- `gamma`: Factor de descuento asociado a la recompensa.
- `epsilon`, `epsilon_end`, `epsilon_decay`: Parámetros para la implementación de la política ϵ-greedy. Representan el valor inicial, final y el decaímiento asociados a ϵ.
- `mem_size`: Cantidad de experiencias almacenadas en la memoria.
- `batch_size`: Cantidad de experiencias que se recuperan en cada instancia de aprendizaje para actualizar la red.

Dentro de la clase agente, además se encuentran los métodos y funciones asociados al aprendizaje en sí, es decir, la creación y actualización de la red. Para implementar la misma, se utiliza la librería `keras`.

5.3 Resultados obtenidos.

Se ejecutó el algoritmo para una cadena de largo 7. Se muestran los resultados para la fidelidad máxima alcanzada, la recompensa asociada y el tiempo en el que se logra en las figuras 5.3, 5.4 y 5.5, respectivamente. Se grafican los resultados para cada episodio (amarillo) junto con el promedio tomado sobre 100 episodios (líneas continuas). Los parámetros utilizados se detallan en la tabla 5.1 y, exceptuando la frecuencia de aprendizaje, coinciden con los del trabajo [19]. Además, en la figura 5.6 se muestra la evolución del parámetro epsilon a lo largo de los episodios.

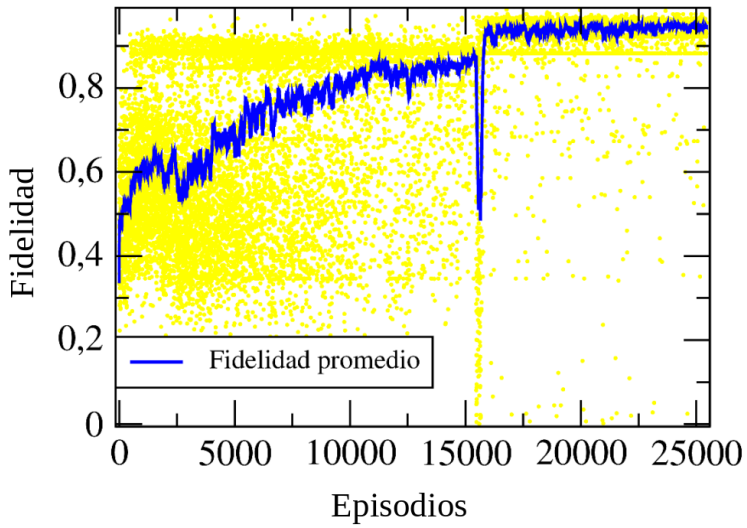

Figura 5.3: Fidelidad máxima en función del número de episodios.

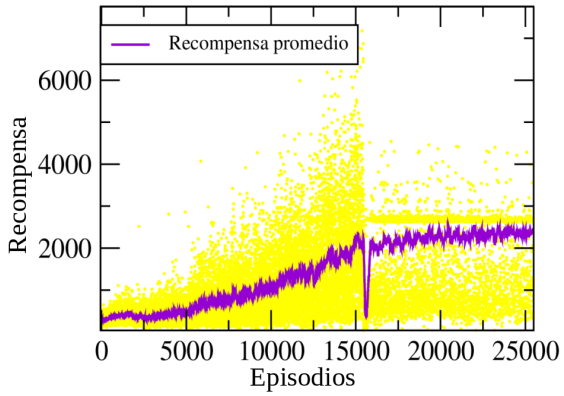

Figura 5.4: Crecimiento de la recompensa en función del número de episodios.

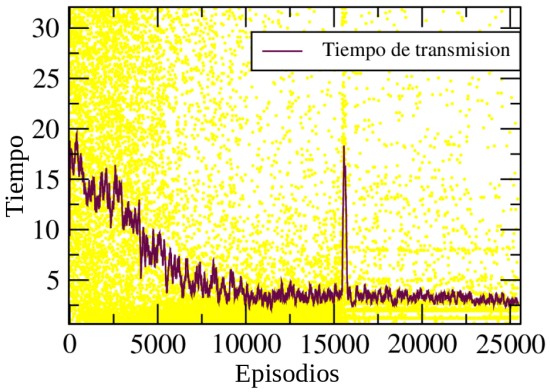

Figura 5.5: Tiempo en el que el sistema alcanza la fidelidad máxima de transmisión.

Se puede ver que, efectivamente, el agente logra encontrar una política que le permite efectuar la secuencia correcta de acciones para lograr una transmisión con más del 95 % de fidelidad. El método converge aproximadamente luego de 15000 episodios. Resulta notorio que, justo antes de alcanzar la convergencia, ocurre un "pico" en los valores de fidelidad, recompensa e (inversamente) tiempo de transmisión. Si bien la aparición de estas irregularidades es común en los procesos de aprendizaje, es interesante ver cómo luego de explorar las acciones menos favorables el agente encuentra la política óptima.

Parámetro	Valor
Tamaño del minibatch	32
Tamaño de la memoria	40000
Tasa de aprendizaje (α)	0.01
Dec. de la recompensa (γ)	0.95
$N°$ de capas ocultas	2
Neuronas por capa oculta	120
Epsilon inicial (ϵ)	1
Epsilon final	0.01
Frecuencia de aprendizaje	1
Tolerancia (1-F) (ζ)	0.05
Tiempo máximo (τ)	32
Paso temporal (dt)	0.15
Int. de acoplamientos (J)	1
Campo externo (B)	100

Tabla 5.1: Parámetros (del sistema) e hiperparámetros (del algoritmo de aprendizaje) utilizados.

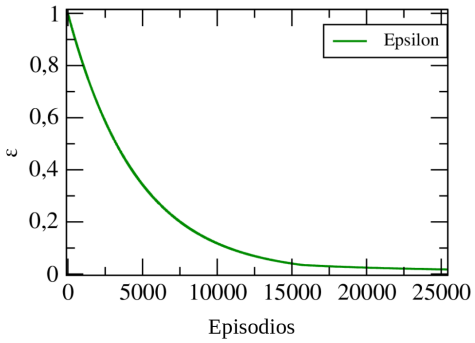

Figura 5.6: Variación del valor de ϵ a lo largo de los distintos episodios. Se puede ver que a medida que el algoritmo avanza, este decrece aumentando así la probabilidad de que el agente base sus decisiones en experiencias previas.

Como se observa en la figura 5.5, el tiempo de transmisión es cada vez menor. Esto tiene sentido al considerar el efecto del factor de descuento γ en la ecuación (3.2) para la recompensa total R. Más allá de la recompensa instantánea (5.5), el agente busca secuencias de acciones que permitan que la transmisión se realice en menos tiempo para maximizar R. Finalmente, logra encontrar la política que no solo maximiza $F(t)$ sino que también minimiza el tiempo de transmisión.

5.3.1 Dependencia de los parámetros del sistema.

Si bien no es posible explicar de manera determinista el efecto de los parámetros del sistema sobre el proceso de aprendizaje, se realizaron algunas variaciones sobre los mismos para explorar sus efectos. Los resultados obtenidos para la fidelidad se muestran en la figura 5.7 y se comparan con la curva de la figura 5.3. Para estudiar mejor la convergencia, se efectuaron 50000 episodios de aprendizaje.

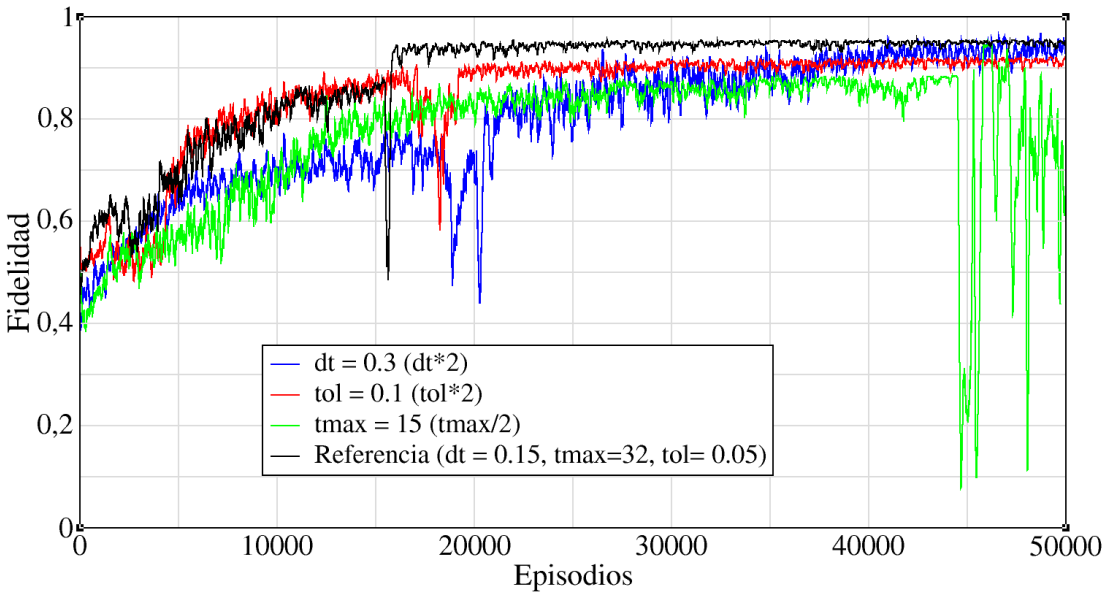

Figura 5.7: Fidelidad obtenida en función de los episodios de aprendizaje. Nuevamente, se toman los promedios sobre 100 episodios. En color negro, la curva con los parámetros de referencia de la figura 5.3. Luego, se muestran los resultados obtenidos al duplicar los intervalos temporales (azul), duplicar la tolerancia preestablecida (rojo) y tomar la mitad del tiempo máximo (verde).

En primer lugar, se observa que al acortar el tiempo máximo de evolución, el algoritmo se vuelve inestable antes de alcanzar la convergencia. Esto sugiere que, si bien al encontrar la política óptima el tiempo de transmisión resulta mucho menor al máximo, es necesario para el agente explorar acciones a tiempos largos para llegar a la misma. Por otro lado, al tomar intervalos de tiempo más largos, al algoritmo le toma más episodios converger. En principio, esto podría estar relacionado al hecho de que, al realizar la mitad de pasos temporales, también se reduce a la mitad el número de instancias de aprendizaje. Con esta cantidad de episodios, no fue posible ver si el algoritmo realmente converge. De todas

formas, la tendencia no sugiere irregularidades del tipo de las que se presentan en el caso de la variación del tiempo máximo. Finalmente, se puede ver que aumentar el valor de la tolerancia hace que el algoritmo converja al nuevo valor establecido. Sin embargo, se está sacrificando un 0.5 % de fidelidad sin obtener una mejora respecto al número de episodios necesario.

El efecto de los hiper-parámetros asociados al algoritmo de aprendizaje, como la tasa de aprendizaje α o el factor de descuento γ, es significativo pero continúa siendo un problema abierto [36]. Por ahora, no se realizaron experimentos variando los mismos en el marco de este trabajo. Sin embargo, dado que estos resultados permitieron verificar la utilidad del *DRL* como técnica de optimización, es una de las líneas de investigación que se prevé tomar a futuro. En general, los resultados son coherentes con los obtenidos por en el trabajo de referencia [19]. Independientemente de los resultados para esta cadena particular, el objetivo de esta etapa era verificar la posibilidad de aplicar este tipo de aprendizaje a problemas físicos, en particular, a la transmisión de estados. Se considera que esta tarea fue realizada exitosamente, sentando un precedente favorable para la exploración futura de esta técnica y su aplicación a sistemas cuánticos.

Parte III

Discusión y Conclusiones

Discusión y Conclusiones

Respecto al algoritmo genético

En general, el algoritmo genético permite encontrar buenos resultados siendo su mayor ventaja una alta rapidez de cómputo en comparación a otros métodos. Para el Hamiltoniano XX, se encontraron acoplamientos que permitían fidelidades de más del 99 % para cadenas de hasta 70 espines en minutos. Otra característica interesante del método es su simplicidad. Fue posible encontrar valores que maximizan la fidelidad de transmisión sin introducir, a priori, mucha información física. Aun así, los resultados lograban reproducir en gran manera el comportamiento físico conocido. Por ejemplo, se optimizó el sistema encontrando un espectro ordenado, incluso cuando, en una primera instancia, esta característica no se incluía como algo a considerar en la aptitud.

Al estudiar el Hamiltoniano de Heisenberg, se corroboró el aumento en la complejidad de optimización para ese sistema. Fue posible observar la rugosidad del espacio de optimización y como el problema se complica tanto al aumentar el largo de las cadenas como al pasar de un sistema a otro, aumentando el valor del parámetro Δ de anisotropía. Se encontró que para este modelo, al tratarse de una optimización más compleja, resulta más relevante el papel de las funciones asociadas al algoritmo. En especial, se observó que utilizar una función mutación personalizada permite suavizar en gran medida las soluciones. Si bien implican un pequeño sacrificio en la velocidad de cómputo, permiten alcanzar valores mayores y más estables de fidelidad.

En el futuro, se prevé realizar un mayor número de experimentos para encontrar funciones tanto mutación como *fitness* que se adecúen mejor al problema. Entre las posibilidades, se ha planteado utilizar mutaciones de tipo adaptativas, es decir, que cambien su forma y efectos de acuerdo al sistema y su comportamiento. Una posibilidad sería que se apliquen cambios mayores o menores según la convergencia o no de la probabilidad de transición al valor esperado. Se ha propuesto, también, la combinación de las distintas funciones *fitness* con parámetros adaptativos. Por ejemplo, utilizar pesos entre distintas funciones de aptitud y añadirlos como genes, dejando de esta forma que el algoritmo encuentre automáticamente que función resulta más conveniente.

Respecto al modelo de *DRL*

Fue posible reproducir exitosamente algunos de los resultados del trabajo de referencia [19]. En particular, se verificó que puede entrenarse un modelo de redes neuronales

para optimizar el problema de transferencia de estados. El agente logra encontrar una secuencia óptima de pulsos magnéticos que le permiten enviar un estado de una punta de la cadena a otra con un 95 % de fidelidad. Puede observarse el aumento respectivo de la recompensa. También, se muestra que, una vez entrenado el modelo, los tiempos de transmisión alcanzan valores del orden previsto según el QSL, es decir, N/2.

En este trabajo el objetivo principal fue desarrollar los códigos y verificar los resultados previos. A futuro, sería interesante poder estudiar los efectos de los hiper-parámetros o incluso implementar variaciones del algoritmo utilizando, por ejemplo, otras arquitecturas de red. El mayor freno para realizar una mayor cantidad y variedad de experimentos fue el tiempo de cómputo. Sin embargo, actualmente, se están adaptando los códigos para poder ejecutarlos utilizando GPU, lo cual aumentará ampliamente la velocidad de obtención de resultados.

Conclusiones generales.

La implementación de métodos de aprendizaje automático a problemas físicos resulta prometedora. En particular, fue posible ver que se pueden obtener buenos resultados en el contexto del problema de interés. En el caso del algoritmo genético, se destaca la rapidez computacional ya que se logra optimizar funciones con dependencia de una gran cantidad de variables en tiempos considerablemente cortos.

Como comentario final, es interesante plantear la discusión sobre la posibilidad de encontrar resultados, en principio, satisfactorios, sin conocer en profundidad la física del problema. Esto resulta muy conveniente, en especial, para estudiar sistemas con propiedades desconocidas. Por supuesto, este tipo de algoritmos no necesariamente permitirían descubrir dichas propiedades. Sin embargo, si lo que se busca es que un sistema realice una tarea específica, no hace falta utilizar métodos de optimización que provean información sobre otros aspectos. Por ejemplo, en este caso, se busca modelar un sistema de forma que se maximice la fidelidad en la transmisión, entonces, no resulta relevante un análisis de los autoestados como el que proveerían, entre otros, los métodos de tipo variacional. En cambio, puede ser preferible utilizar métodos, como los estudiados en este trabajo, que al optimizar priorizan que se realice la tarea de interés. Aún más, al no depender de propiedades particulares, podrían ser aplicables a una variedad más amplia de sistemas.

Apéndice A

Forma explícita de los Hamiltonianos utilizados.

En este apéndice se detallan los cálculos necesarios para obtener las formas explícitas de los Hamiltonianos XX y Heisenberg en la base de una excitación. Luego, se detallan las matrices de las 16 acciones posibles asociadas al Hamiltoniano con pulsos magnéticos en la misma base.

A.1 Hamiltoniano XX

Para calcular, la forma explícita de este Hamiltoniano es necesario observar cómo actúa sobre los estados de la base de una excitación. Es decir, se tienen los estados dados por:

$$|j\rangle \;=\; |\downarrow\downarrow\downarrow\downarrow\;\ldots\uparrow\downarrow\downarrow\downarrow\downarrow\downarrow\rangle \tag{A.1}$$

Y se quiere ver cómo actúa el Hamiltoniano XX:

$$H_{xx} = -\sum_{i=1}^{N} J_{i,i+1}\;\left(\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y\right) \tag{A.2}$$

Para ello, se reescribe el mismo utilizando operadores escalera o "*ladder*":

$$\sigma_i^x \;=\; \frac{\sigma_i^+ + \sigma_i^-}{2} \quad ,\; \sigma_i^y \;=\; \frac{\sigma_i^+ - \sigma_i^-}{2i} \tag{A.3}$$

Al reemplazar estas expresiones para las matrices de Pauli en (A.2), se obtiene:

$$H_{xx} = -\sum_{i=1}^{N} J_{i,i+1}\left\{\left(\frac{\sigma_i^+ + \sigma_i^-}{2}\right)\left(\frac{\sigma_{i+1}^+ + \sigma_{i+1}^-}{2}\right) - \left(\frac{\sigma_i^+ - \sigma_i^-}{2}\right)\left(\frac{\sigma_{i+1}^+ - \sigma_{i+1}^-}{2}\right)\right\} \tag{A.4}$$

Que al distribuir resulta en:

$$= -\frac{1}{2}\sum_{i=1}^{N} J_{i,i+1}\left\{\sigma_i^+\sigma_{i+1}^- + \sigma_i^-\sigma_{i+1}^+\;\right\} \tag{A.5}$$

Ahora, se aplica este Hamiltoniano sobre el j-ésimo estado:

$$\Longrightarrow H_{xx}\,|\downarrow\downarrow\ ...\uparrow\downarrow\downarrow\downarrow\downarrow\downarrow\rangle = \left(-\frac{1}{2}\sum_{i=1}^{N} J_{i,i+1}\left\{\sigma_i^+\sigma_{i+1}^- + \sigma_i^-\sigma_{i+1}^+\right\}\right)|\downarrow\ ...\uparrow\downarrow\downarrow\downarrow\downarrow\downarrow\rangle$$

Para ver cómo actúa, se estudia en primer lugar la acción del primer término:

$$\sum_{i=1}^{N} J_{i,i+1}\ \ \sigma_i^+\sigma_{i+1}^-\,|\downarrow\downarrow\ ...\uparrow\ \downarrow\downarrow\downarrow\downarrow\downarrow\rangle = \sum_{i=1}^{N} J_{i,i+1}\ \ \delta_{i+1,j}\sigma_i^+\,|\downarrow\downarrow\ ...\downarrow\ \downarrow\downarrow\downarrow\downarrow\downarrow\rangle =$$

$$J_{j-1,j}\ \ \sigma_{j-1}^+\,|\downarrow\downarrow\ ...\downarrow\ \downarrow\downarrow\downarrow\downarrow\downarrow\rangle = |\downarrow\downarrow\ ...\uparrow\ \downarrow\ \downarrow\downarrow\downarrow\downarrow\downarrow\rangle =$$

$$J_{j-1,j}\,|j-1\rangle$$

Teniendo en cuenta que los operadores correspondientes a distintos espacios conmutan, se calcula la acción del segundo término de manera análoga:

$$\left(\sum_{i=1}^{N} J_{i,i+1}\ \ \sigma_i^-\sigma_{i+1}^+\right)|\downarrow\downarrow\ ...\uparrow\ \downarrow\downarrow\downarrow\downarrow\downarrow\rangle = J_{i,i+1}\ \ \left(\sum_{i=1}^{N}\sigma_{i+1}^+\sigma_i^-\right)|\downarrow\downarrow\ ...\uparrow\ \downarrow\downarrow\downarrow\downarrow\downarrow\rangle =$$

$$\sum_{i=1}^{N} J_{i,i+1}\ \ \sigma_{i+1}^+\,\delta_{ij}\,|\downarrow\downarrow\ ...\downarrow\ \downarrow\downarrow\downarrow\downarrow\downarrow\rangle\ \ =\ \ J_{j,j+1}\ \sigma_{j+1}^+\ \ |\downarrow\downarrow\ ...\downarrow\ \downarrow\downarrow\downarrow\downarrow\downarrow\rangle =$$

$$J_{j,j+1}\,|j+1\rangle$$

Finalmente, se tiene que:

$$H_{xx}\ \ |j\rangle\ \ =\ \ -\frac{1}{2}\left(J_{j-1,j}\,|j-1\rangle + J_{j,j+1}\,|j+1\rangle\right) \tag{A.6}$$

Entonces, en la base de una excitación la forma explícita de los elementos de matriz estará dada por:

$$\begin{gathered}\langle i|H|j\rangle\ \ = -\ \tfrac{1}{2}\ \ \langle i|\ \ \left(J_{j-1,j}\,|j-1\rangle + J_{j,j+1}\,|j+1\rangle\right)\ \ = \\ -\tfrac{1}{2}\ \left(J_{j-1,j}\ \delta_{i,j-1} + J_{j,j+1}\ \delta_{i,j+1}\right)\end{gathered} \tag{A.7}$$

Finalmente, la matriz completa resulta:

$$-\frac{1}{2}\begin{bmatrix} 0 & J_{1,2} & & ... & 0 \\ J_{1,2} & 0 & J_{2,3} & ... & 0 \\ 0 & J_{2,3} & 0 & ... & \vdots \\ 0 & \vdots & \vdots & \ddots & J_{N-1,N} \\ 0 & 0 & 0 & J_{N-1,N} & 0 \end{bmatrix} \tag{A.8}$$

A.2 Hamiltoniano de Heisenberg

Para encontrar la forma explícita del Hamiltoniano de Heisenberg, solo es necesario analizar el tercer término ya que los otros dos se corresponden al H_{xx}. Luego, se reescribe según:

$$H_{Heis} \;=\; -\sum_{i=1}^{N} J_{i,i+1}\left(\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y + \;\sigma_i^z\sigma_{i+1}^z\right) = H_{xx} - \sum_{i=1}^{N} J_{i,i+1}\;\;\sigma_i^z\sigma_{i+1}^z \tag{A.9}$$

Para trabajar de forma más simple, se define:

$$H_z \;=\; -\sum_{k=1}^{N} J_{k,k+1}\;\sigma_k^z\sigma_{k+1}^z$$

$$H_z\,|j\rangle \;=\; -\sum_{k=1}^{N} J_{k,k+1}\;\sigma_k^z\sigma_{k+1}^z\,|j\rangle \;=$$

Se estudia cómo actúa sobre el j-ésimo estado:

$$\sigma_k^z\sigma_{k+1}^z\,|j\rangle \implies \sigma_{k+1}^z\,|j\rangle \;= \begin{cases} \frac{1}{2}\,|j\rangle & k+1=j \\ -\frac{1}{2}\,|j\rangle & k+1\neq\;j \end{cases}$$

$$\implies \sigma_k^z\,|j\rangle \;= \begin{cases} \frac{1}{2}\,|j\rangle & k=j \\ -\frac{1}{2}\,|j\rangle & k\neq\;j \end{cases}$$

$$J_{k,k+1}\;\sigma_k^z\sigma_{k+1}^z\,|j\rangle \;=\; \begin{cases} -\frac{1}{4}\,|j\rangle & j=k,\;k+1 \\ \\ \frac{1}{4}\,|j\rangle & j\neq k,\;k+1 \end{cases} \;=\; \frac{1}{4}J_{k,k+1}(-\delta_{j,k}-\delta_{j,k+1}+(1-\delta_{j,k}-\delta_{j,k+1}))\,|j\rangle$$

Se puede ver, entonces, que este término extra tendrá efecto únicamente sobre la diagonal que puede calcularse según:

$$H^z_{jj} = -\frac{1}{4}\ \langle j|\ \sum_{k=1}^{N} J_{k,k+1}(-\delta_{j,k} - \delta_{j,k+1} + (1 - \delta_{j,k} - \delta_{j,k+1}))\,|j\rangle$$

$$-\frac{1}{4}\ \langle j|\ \sum_{k=1}^{N} J_{k,k+1}(1 - 2\delta_{j,k} - 2\delta_{j,k+1})\,|j\rangle$$

Finalmente, los elementos de la diagonal resultan:

$$H^z_{jj} = -\frac{1}{4}\ \left(\sum_{k=1}^{N} J_{k,k+1} - 2J_{j,j+1} - 2J_{j-1,j}\right)$$

Combinando esto con lo calculado para el Hamiltoniano XX, se obtiene:

$$\begin{bmatrix} H^z_{jj} & -\frac{J_{1,2}}{2} & ... & ... & ... & ... \\ -\frac{J_{1,2}}{2} & H^z_{jj} & -\frac{J_{2,3}}{2} & ... & ... & ... \\ 0 & -\frac{J_{2,3}}{2} & ... & ... & ... & ... \\ 0 & .. & 0 & -\frac{J_{N-2,N-1}}{2} & H^z_{jj} & -\frac{J_{N-1,N}}{2} \\ 0 & .. & .. & 0 & -\frac{J_{N-1,N}}{2} & H^z_{jj} \end{bmatrix} \tag{A.10}$$

A.3 Hamiltonianos de las matrices de las 16 acciones posibles para el algoritmo de *DRL*.

Se busca encontrar la forma explícita de las matrices asociadas a las 16 acciones definidas según (5.1) cuya forma general es:

$$H(t)\ =\ \sum_{i=1}^{N-1} \frac{-J}{2}\left(\sigma^i_x\sigma^{i+1}_x\ +\ \sigma^i_y\sigma^{i+1}_y\right) +\ \sum_{k=1}^{N} B_k(t)\sigma^k_z$$

Nuevamente, como los primeros términos se corresponden con el Hamiltoniano XX, sólo es necesario encontrar una expresión para el último. La acción del operador σ^k_z sobre los estados de la base está dada por:

$$\sigma^k_z\,|j\rangle = \begin{cases} \frac{1}{2}\,|j\rangle & j = k \\ -\frac{1}{2}\,|j\rangle & j \neq k \end{cases} \tag{A.11}$$

Es decir,

$$\sigma^k_z\,|j\rangle = \frac{1}{2}\delta_{j,k} + (1 - \delta_{j,k})(-\frac{1}{2})\,|j\rangle = \left(\delta_{j,k} - \frac{1}{2}\right) \tag{A.12}$$

De esta forma, las 16 matrices estarán dadas por:

- $a_t \;=\; 1 \implies$ Encender B_1: $\quad \langle i|H_{xx} + B\ \sigma_z^1|j\rangle \;=\; [H_{xx}] \;+\; B\ \delta_{ji}\ (\delta_{1i} - 1/2)$

$$M_1 = \begin{bmatrix} B/2 & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & -B/2 & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & -B/2 & ... & 0 & .... & \vdots \\ 0 & \vdots & \vdots & \ddots & & & 0 \\ \vdots & \vdots & \vdots & \vdots & -B/2 & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & -B/2 & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & -B/2 \end{bmatrix} \tag{A.13}$$

- $a_t \;=\; 2 \implies$ Encender B_2: $\quad \langle i|H_{xx} + B\ \sigma_z^2|j\rangle \;=\; [H_{xx}] \;+\; B\ \delta_{ji}\ (\delta_{2i} - 1/2)$

$$M_2 = \begin{bmatrix} -B/2 & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & B/2 & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & -B/2 & ... & 0 & .... & \vdots \\ 0 & \vdots & \vdots & \ddots & & & 0 \\ \vdots & \vdots & \vdots & \vdots & -B/2 & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & -B/2 & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & -B/2 \end{bmatrix} \tag{A.14}$$

- $a_t \;=\; 3 \implies$ Encender B_1 y B_2: $M_1 + M_2$

$$M_3 = \begin{bmatrix} 0 & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & 0 & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & -B & ... & 0 & .... & \vdots \\ 0 & \vdots & \vdots & \ddots & & & 0 \\ \vdots & \vdots & \vdots & \vdots & -B & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & -B & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & -B \end{bmatrix} \tag{A.15}$$

- $a_t \;=\; 4 \implies$ Encender B_3: $\quad \langle i|H_{xx} + B\ \sigma_z^3|j\rangle \;=\; [H_{xx}] \;+\; B\ \delta_{ji}\ (\delta_{3i} - 1/2)$

$$M_4 = \begin{bmatrix} -B/2 & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & -B/2 & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & B/2 & ... & 0 & .... & \vdots \\ 0 & \vdots & \vdots & \ddots & & & 0 \\ \vdots & \vdots & \vdots & \vdots & -B/2 & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & -B/2 & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & -B/2 \end{bmatrix} \tag{A.16}$$

- $a_t = 5 \implies$ Encender B_1 y B_3: $M_1 + M_4$

$$M_5 = \begin{bmatrix} 0 & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & -B & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & 0 & ... & 0 & .... & \vdots \\ 0 & \vdots & \vdots & \ddots & & & 0 \\ \vdots & \vdots & \vdots & \vdots & -B & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & -B & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & -B \end{bmatrix} \tag{A.17}$$

- $a_t = 6 \implies$ Encender B_2 y B_3: $M_2 + M_3$

$$M_6 = \begin{bmatrix} -B & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & 0 & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & 0 & ... & 0 & .... & \vdots \\ 0 & \vdots & \vdots & \ddots & & & 0 \\ \vdots & \vdots & \vdots & \vdots & -B & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & -B & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & -B \end{bmatrix} \tag{A.18}$$

- $a_t = 7 \implies$ Encender B_1, B_2 y B_3: $M_1 + M_2 + M_4$

$$M_7 = \begin{bmatrix} -B/2 & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & -B/2 & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & -B/2 & ... & 0 & .... & \vdots \\ 0 & \vdots & \vdots & \ddots & & & 0 \\ \vdots & \vdots & \vdots & \vdots & -3B/2 & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & -3B/2 & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & -3B/2 \end{bmatrix} \tag{A.19}$$

- $a_t = 8 \implies$ Encender B_{N_2}: $\left\langle i \middle| H_{xx} + B\ \sigma_z^{N-2} \middle| j \right\rangle = [H_{xx}] + B\ \delta_{ji}\ (\delta_{(N-2)i} - 1/2)$

$$M_8 = \begin{bmatrix} -B/2 & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & -B/2 & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & -B/2 & ... & 0 & \vdots & \vdots \\ 0 & \vdots & \vdots & \ddots & \vdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & B/2 & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & -B/2 & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & -B/2 \end{bmatrix} \tag{A.20}$$

- $a_t = 9 \Longrightarrow$ Encender B_{N_1}: $\left\langle i \middle| H_{xx} + B\ \sigma_z^{N-1} \middle| j \right\rangle = [H_{xx}] + B\ \delta_{ji}\ (\delta_{(N-1)i} - 1/2)$

$$M_9 = \begin{bmatrix} -B/2 & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & -B/2 & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & -B/2 & ... & 0 & \vdots & \vdots \\ 0 & \vdots & \vdots & \ddots & \vdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & -B/2 & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & B/2 & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & -B/2 \end{bmatrix} \tag{A.21}$$

- $a_t = 10 \Longrightarrow$ Encender B_{N_2} y B_{N_1}: $M_8 + M_9$

$$M_{10} = \begin{bmatrix} -B & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & -B & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & -B & ... & 0 & \vdots & \vdots \\ 0 & \vdots & \vdots & \ddots & \vdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & 0 & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & 0 & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & -B \end{bmatrix} \tag{A.22}$$

- $a_t = 11 \Longrightarrow$ Encender B_N: $\left\langle i \middle| H_{xx} + B\ \sigma_z^{N} \middle| j \right\rangle = [H_{xx}] + B\ \delta_{ji}\ (\delta_{(N)i} - 1/2)$

$$M_{11} = \begin{bmatrix} -B/2 & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & -B/2 & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & -B/2 & ... & 0 & \vdots & \vdots \\ 0 & \vdots & \vdots & \ddots & \vdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & -B/2 & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & -B/2 & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & B/2 \end{bmatrix} \tag{A.23}$$

- $a_t = 12 \Longrightarrow$ Encender B_N y B_{N-2} : $M_8 + M_{11}$

$$M_{12} = \begin{bmatrix} -B & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & -B & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & -B & ... & 0 & \vdots & \vdots \\ 0 & \vdots & \vdots & \ddots & \vdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & 0 & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & -B & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & 0 \end{bmatrix} \tag{A.24}$$

- $a_t = 13 \implies$ Encender B_N y B_{N-1} : $M_9 + M_{11}$

$$M_{13} = \begin{bmatrix} -B & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & -B & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & -B & ... & 0 & \vdots & \vdots \\ 0 & \vdots & \vdots & \ddots & \vdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & -B & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & 0 & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & 0 \end{bmatrix} \tag{A.25}$$

- $a_t = 14 \implies$ Encender B_N, B_{N-1} y B_{N-2} : $M_8 + M_9 + M_{11}$

$$M_{14} = \begin{bmatrix} -3B/2 & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & -3B/2 & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & -3B/2 & ... & 0 & \vdots & \vdots \\ 0 & \vdots & \vdots & \ddots & \vdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & -B/2 & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & -B/2 & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & -B/2 \end{bmatrix} \tag{A.26}$$

- $a_t = 15 \implies$ Todos encendidos : $M_1 + M_2 + M_4 + M_8 + M_9 + M_{11}$

$$M_{15} = \begin{bmatrix} -2B & J_{1,2} & 0 & 0 & ... & ... & 0 \\ J_{1,2} & -2B & J_{2,3} & ... & 0 & ... & 0 \\ 0 & J_{2,3} & -2B & ... & 0 & .... & \vdots \\ 0 & \vdots & \vdots & \ddots & & & 0 \\ \vdots & \vdots & \vdots & \vdots & -2B & J_{N-2,N-1} & 0 \\ 0 & 0 & 0 & 0 & J_{N-2,N-1} & -2B & J_{N-1,N} \\ 0 & 0 & 0 & 0 & 0 & J_{N-1,N} & -2B \end{bmatrix} \tag{A.27}$$

Bibliografía

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2019, ISBN: 9781107002173.

[2] D. P. DiVincenzo, "The physical implementation of quantum computation," *Fortschritte Der Physik*, vol. 48, no. 9-11, pp. 771–783, Sep. 2000. DOI: https://doi.org/10.1002/1521-3978(200009)48:9/11%3C771::AID-PROP771%3E3.0.CO;2-E.

[3] S. Bose, "Quantum communication through an unmodulated spin chain," *Physical Review Letters*, vol. 91, no. 207901, 2003. DOI: 10.1103/physrevlett.91.207901.

[4] G. M. Nikolopoulos and I. Jex, *Quantum State Transfer and Network Engineering*. Berlin Springer, 2016, ISBN: 9783662511022.

[5] G. Carleo and M. Troyer, "Solving the quantum many-body problem with artificial neural networks," *Science*, vol. 355, no. 6325, pp. 602–606, Feb. 2017. DOI: 10.1126/science.aag2302.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015. DOI: 10.1038/nature14539.

[7] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan. 2016. DOI: 10.1038/nature16961.

[8] A.-B. M. Salem, K. Revett, and E.-S. A. El-Dahshan, "Machine learning in electrocardiogram diagnosis," in *2009 International Multiconference on Computer Science and Information Technology*, 2009, pp. 429–433. DOI: 10.1109/IMCSIT.2009.5352689.

[9] B. Mohammed, I. Awan, H. Ugail, and M. Younas, "Failure prediction using machine learning in a virtualised hpc system and application," *Cluster Computing*, vol. 22, no. 2, pp. 471–485, Mar. 2019. DOI: 10.1007/s10586-019-02917-1.

[10] C. Zhang and Y. Zhang, "Songnet: Real-time music classification," 2018. [Online]. Available: https://cs229.stanford.edu/proj2018/report/53.pdf.

[11] P. Serra, A. Ferrón, and O. Osenda, "Almost perfect quantum state transfer on isotropic and anisotropic heisenberg spin chains with tailored site dependent exchange couplings," *Physics Letters A*, vol. 449, no. 128362, Oct. 2022. DOI: https://doi.org/10.1016/j.physleta.2022.128362.

[12] D. Acosta Coden, S. Gómez, A. Ferrón, and O. Osenda, "Controlled quantum state transfer in xx spin chains at the quantum speed limit," *Physics Letters A*, vol. 387, no. 127009, Jan. 2021. DOI: 10.1016/j.physleta.2020.127009.

[13] R. J. Baxter, *Exactly Solved Models in Statistical Mechanics*. Academic Press, 1982, ISBN: 0120831805.

[14] A. Klümper and J. Zittartz, "The eight-vertex model: Spectrum of the transfer matrix and classification of the excited states," *Zeitschrift für Physik B Condensed Matter*, vol. 75, pp. 371–384, Sep. 1989. DOI: https://doi.org/10.1007/BF01321825.

[15] S. Bose, "Quantum communication through spin chain dynamics: An introductory overview," *Contemporary Physics*, vol. 48, no. 1, pp. 13–30, Jan. 2007. DOI: 10.1080/00107510701342313.

[16] H. L. Haselgrove, "Optimal state encoding for quantum walks and quantum communication over spin systems," *Physical Review A*, vol. 72, no. 062326, Dec. 2005. DOI: 10.1103/physreva.72.062326.

[17] M. Christandl, N. Datta, A. Ekert, and A. J. Landahl, "Perfect state transfer in quantum spin networks," *Physical Review Letters*, vol. 92, no. 187902, May 2004. DOI: 10.1103/physrevlett.92.187902.

[18] A. Zwick and O. Osenda, "Quantum state transfer in a xx chain with impurities," *Journal of Physics A: Mathematical and Theoretical*, vol. 44, no. 105302, Feb. 2011. DOI: 10.1088/1751-8113/44/10/105302.

[19] X.-M. Zhang, Z.-W. Cui, X. Wang, and M.-H. Yung, "Automatic spin-chain learning to explore the quantum speed limit," *Physical Review A*, vol. 97, no. 052333, May 2018. DOI: 10.1103/physreva.97.052333.

[20] A. Kay, "The limits of quantum state transfer for field-free heisenberg chains," Jul. 2022. [Online]. Available: https://arxiv.org/abs/1906.06223.

[21] A. Ferrón, P. Serra, and O. Osenda, "Understanding the propagation of excitations in quantum spin chains with different kind of interactions," *Physica Scripta*, vol. 97, no. 115103, Oct. 2022. DOI: 10.1088/1402-4896/ac955c.

[22] S. Sachdev, *Quantum Phase Transitions*. Cambridge University Press, Apr. 2011, ISBN: 9781139500210.

[23] T. Caneva, M. Murphy, T. Calarco, *et al.*, "Optimal control at the quantum speed limit," *Physical Review Letters*, vol. 103, no. 240501, Dec. 2009. DOI: 10.1103/physrevlett.103.240501.

[24] M. J. Kochnderfer and T. A. Wheeler, *Algorithms for Optimization*. The Mit Press, 2019, ISBN: 9780262039420.

[25] S. Mirjalili, *Evolutionary algorithms and neural networks : theory and applications*. Cham Springer, 2019, ISBN: 9783319930244.

[26] A. F. Gad, *Pygad: An intuitive genetic algorithm python library*, 2021. [Online]. Available: https://arxiv.org/abs/2106.06158.

[27] V. M. Flugsurd, "Solving quantum mechanical problems with machine learning," Ph.D. dissertation, Faculty of Mathematics and Natural Sciences University of Oslo, Jun. 2018. [Online]. Available: https://www.duo.uio.no/bitstream/handle/10852/65997/5/thesis.pdf.

[28] S. Saito, Y. Wenzhuo, and R. Shanmugamani, *Python reinforcement Learning Projects : Eight hands-on Projects Exploring Reinforcement Learning Algorithms Using TensorFlow*. Packt Publishing Ltd, 2018, ISBN: 9781788991612.

[29] R. S. Sutton and A. Barto, *Reinforcement learning : an introduction*, 2nd ed. The MIT Press, 2018, ISBN: 9780262039246.

[30] S. Ravichandiran, *Hands-on Reinforcement Learning with Python : Master Reinforcement and Deep Reinforcement Learning Using OpenAI Gym and TensorFlow*. Packt Publishing, 2018, ISBN: 9781788836524.

[31] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958. DOI: 10.1037/h0042519.

[32] K. Team, *Keras documentation: Adam*. [Online]. Available: https://keras.io/api/optimizers/adam/.

[33] P. Serra, A. Ferrón, and O. Osenda, "Exact solution of a family of staggered heisenberg chains with conclusive pretty good quantum state transfer," *Journal of Physics A: Mathematical and Theoretical*, vol. 55, no. 405302, Sep. 2022. DOI: 10.1088/1751-8121/ac901d.

[34] P. Tabor, *Reinforcement learning - github: Philtabor*, 2021. [Online]. Available: https://github.com/philtabor/Youtube-Code-Repository.

[35] G. Brockman, V. Cheung, L. Pettersson, *et al.*, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[36] J. C. Barsce, "Optimización bayesiana de híper-parámetros en algoritmos de aprendizaje por refuerzos," Ph.D. dissertation, Universidad Tecnológica Nacional, Facultad Regional Santa Fe, Apr. 2022.