Universidad Internacional de la Rioja (UNIR)

Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Computación Cuántica

Quantum-Annealed Action Selection Policy for Reinforcement Learning

Trabajo Fin de Estudios

Presentado por: Caponera De Cobellis, Romolo Rosario

Dirigido por: Navarro Burgos, Miguel Ángel

Ciudad: Fuengirola

Fecha: Julio de 2023

Índice de Contenidos

Resumen vi

Abstract vii

1. Introducción 1

1.1. Motivación del trabajo . . . 1

1.2. Planteamiento del trabajo . . . 1

1.3. Estructura del trabajo . . . 2

2. Contexto y Estado del Arte 3

2.1. Inteligencia Artificial . . . 3

2.1.1. Contexto histórico de la Inteligencia Artificial . . . 3

2.1.2. Tipos de Inteligencia Artificial . . . 4

2.2. Computación Cuántica . . . 9

2.2.1. Propiedades cuánticas de la materia . . . 9

2.2.2. Notación de Dirac o bra-ket . . . 14

2.2.3. Modelos de computación cuántica . . . 15

3. Objetivos 20

4. Desarrollo del trabajo 21

4.1. Aprendizaje por refuerzo . . . 21

4.2. Q-Learning . . . 22

4.2.1. Procesos de Decisión de Markov y FrozenLake . . . 29

4.3. Exploración vs explotación . . . 31

4.3.1. Épsilon-greedy . . . 32

4.3.2. Softmax . . . 33

4.3.3. Upper Confidence Bound . . . 34

4.4. Descripción de la política de selección de acciones cuántica propuesta: QAASP 36

4.4.1. Problemas QUBO . . . 36

4.4.2. Política de Selección de Acciones como un problema QUBO . . . 38

4.5. Implementación de la política de selección de acciones . . . 41

4.6. Análisis del rendimiento . . . 42

5. Conclusiones y Trabajo Futuro **50**

Índice de Ilustraciones

2.1. Representación de un algoritmo de aprendizaje supervisado entrenado para distinguir gatos de otros animales. Adaptado de [Pickel, 2021]. 6

2.2. Representación de un algoritmo de aprendizaje no supervisado entrenado para agrupar animales en función de si son de tierra, agua o aire. Adaptado de [Pickel, 2021]. 8

2.3. Representación de un algoritmo de aprendizaje por refuerzo. El ratón representa al agente, que debe moverse por el laberinto (entorno) para obtener la recompensa (comida). Extraído de [Das, 2017]. 9

2.4. Experimento del triple polarizador de Dirac utilizando un polarizador horizontal y uno vertical. Extraído de [Caltech, 2023]. 11

2.5. Experimento del triple polarizador de Dirac utilizando un polarizador intermedio inclinado a 45°. Extraído de [Caltech, 2023]. 12

2.6. Visualización del efecto túnel. La barrera de potencial representa una de las paredes de la caja. Se aprecia cómo la función de onda es mucho más amplia en el interior de la caja (izquierda de la pared) que en el exterior, pero nunca se anula por completo. Extraído de [Trixler, 2013]. 13

2.7. Representación de quantum annealing basado en el efecto túnel (2.7a) y en computación adiabática (2.7b). Adaptado de [Cervera-Lierta, 2018]. 18

4.1. Representación de la recompensa obtenida con una tasa de aprendizaje demasiado alta. Se puede ver cómo, a partir de la segunda iteración, el agente empieza a oscilar alrededor del máximo, sin alcanzarlo. 25

4.2. Representación de la recompensa obtenida con una tasa de aprendizaje demasiado baja. Se puede ver cómo el agente se acerca muy lentamente al punto máximo, necesitando un gran número de iteraciones para completar el entrenamiento. 26

4.3. Representación de la recompensa obtenida con una tasa de aprendizaje óptima. Se puede ver cómo el agente alcanza el máximo en pocos pasos. 27

4.4. Representación pixel-art del entorno FrozenLake. Extraído de [Farama, 2022]. 31

4.5. Representación del intervalo de confianza de UCB. Adaptado de [GeeksForGeeks, 2020]. 35

4.6. Representación de las probabilidades de cada una de las acciones según el multiplicador m utilizado. Los valores esperados para las acciones son, respectivamente, -1, 1, 0.5 y 0.314. 40
4.7. Diagrama de barras que muestra el rendimiento del algoritmo en el entorno FrozenLake 4x4. 20 ejecuciones. Escala logarítmica. 44
4.8. Diagrama de barras que muestra el rendimiento del algoritmo en el entorno FrozenLake 4x4. 100 ejecuciones. Escala logarítmica. 45
4.9. Gráfico box-and-whiskers que muestra el rendimiento del algoritmo en el entorno FrozenLake 4x4. 100 ejecuciones. Escala logarítmica. 46
4.10. Gráfico box-and-whiskers que muestra el rendimiento del algoritmo en el entorno FrozenLake 4x4. 100 ejecuciones. Escala lineal, sin outliers. 47

Índice de Tablas

4.1. Representación de la matriz Q de un entorno con 4 estados y 4 acciones. No se muestran los valores esperados. 23

Resumen

De entre todos los tipos de machine learning, el aprendizaje por refuerzo es posiblemente el menos estudiado en términos de computación cuántica, si bien tiene potencial para beneficiarse en gran medida del no-determinismo intrínseco de la cuántica. Este trabajo explora el uso del Quantum Annealing en el ámbito del aprendizaje por refuerzo y, concretamente, el Q-Learning, proponiendo una política de selección de acciones basada en el annealing. Se proporciona una revisión de algunas de las políticas clásicas más utilizadas, para permitir una mejor comprensión de las mismas, y posteriormente se compara el rendimiento de la política cuántica QAASP con ellas. Las pruebas se realizan en el entorno FrozenLake ofrecido por Gymnasium (anteriormente conocido como OpenAI Gym). Los resultados indican ventajas potenciales ligadas al uso de QAASP, si bien el rendimiento final es similar a las políticas clásicas en términos de episodios hasta el fin del entrenamiento.

Palabras Clave: Computación cuántica, Quantum Annealing, Q-Learning, Aprendizaje por refuerzo.

Abstract

Reinforcement Learning is, amongst all Machine Learning types, the least studied in Quantum Computing, yet one that could greatly benefit from the quantum non-determinism in order to approach the exploration-exploitation dilemma. This work explores the usage of Quntum Annealing in the field of Reinforcement Learning and Q-Learning, proposing an annealing-based action selection policy. An overview of some of the most popular classical action selecion policies is offered, so that they are better understood, after which a performance comparison between the classical policies and the proposed QAASP is given. Tests and benchmarks are performed on the FrozenLake envoironment offered by Gymnasium (formerly OpenAI Gym). Results show some potential benefits on the usage of QAASP, yet an overall similar performance compared to classical policies in terms of episodes until convergence.

Palabras Clave: Quantum computing, Quantum Annealing, Q-Learning, Reinforcement Learning.

1. Introducción

1.1. Motivación del trabajo

La computación cuántica es una tecnología aún muy joven y, si bien resulta muy prometedora, aún es necesaria mucha investigación en este campo para determinar en qué ámbitos es más útil, y cómo aprovechar su potencial. Uno de los campos en los que se está intentando utilizar esta tecnología es la Inteligencia Artificial (IA), puesto que es una rama que requiere una enorme cantidad de recursos computacionales y que se vería enormemente beneficiada de las ventajas que ofrece la computación cuántica. Incluso hay estudios médicos que han identificado efectos cuánticos en el cerebro humano, apuntando a que, quizás, la computación cuántica sea la manera más efectiva de alcanzar una inteligencia artificial superior [Martins, 2022]. En base a estas premisas, el presente trabajo pretende estudiar la aplicabilidad de la computación cuántica en el ámbito de la inteligencia artificial y, más concretamente, en el aprendizaje por refuerzo.

1.2. Planteamiento del trabajo

Este trabajo se centra en el estudio del aprendizaje por refuerzo, y de uno de los algoritmos más populares en este ámbito, el Q-Learning, con el fin de implementar las posibles mejoras que ofrece la computación cuántica. En primer lugar, la naturaleza aleatoria de la cuántica plantea un escenario ideal para resolver el dilema de la exploración-explotación (Ver sección 4.3) de una manera distinta a las utilizadas en la versión clásica del algoritmo, como épsilon-greedy o Softmax. Para ello, se implementará un algoritmo basado en Quantum Annealing para la selección de acciones durante el entrenamiento. Se estudiará por qué este tipo de estrategia puede ser útil en problemas de aprendizaje por refuerzo, y qué aporta respecto al resto de políticas.

Finalmente, se evaluará el rendimiento de este algoritmo cuántico respecto a sus contrapartidas clásicas en términos de episodios hasta la convergencia, utilizando para ello un software específico.

1.3. Estructura del trabajo

En este trabajo se tratan temas de elevada complejidad, como son el Machine Learning o la computación cuántica, sin embargo, aspira a poder ser leído y comprendido por un público general. Por ello, se inicia el trabajo con una introducción a los campos de la inteligencia artificial y la computación cuántica, que buscan sentar una base de conocimientos para la comprensión de apartados posteriores. Más adelante, se detalla la motivación y el funcionamiento de algoritmo Q-Learning, desarrollando los problemas de decisión de Markov, que justifican gran parte de la teoría matemática detrás de este algoritmo. Se hace énfasis en la necesidad de buscar un equilibrio entre la exploración y explotación en este algoritmo, y se justifica cómo la naturaleza cuántica de la materia ofrece una potencial solución a este problema. Una vez introducido esto, se detalla el funcionamiento y la implementación de la política desarrollada, para, acto seguido, realizar una comparación con algunas de las políticas más populares utilizadas en versiones clásicas de este algoritmo. Por último, se presentan las conclusiones de este trabajo, así como posibles líneas de trabajo futuras.

2. Contexto y Estado del Arte

De entre las tecnologías emergentes más influyentes en los últimos tiempos, y más prometedoras en el futuro, dos destacan especialmente por su enorme potencial: la Inteligencia Artificial y la Computación Cuántica. Si bien esta última se encuentra, aún, en un estado muy prematuro, ya se empieza a apreciar el verdadero potencial de esta tecnología, que se plantea como una alternativa a la computación clásica que permitirá aumentar enormemente las capacidades de cómputo disponibles. Por otro lado, la IA es un campo que, si bien está considerablemente más explorado y desarrollado, aún está lejos de alcanzar su máximo esplendor. El presente trabajo se centra en estudiar la posibilidad de combinar ambas tecnologías, tomando los puntos fuertes de cada una, con el objetivo de conseguir un rendimiento superior. Más concretamente, se aplicarán algoritmos cuánticos al Aprendizaje por Refuerzo (Reinforcement Learning, RL). Para contextualizar mejor este trabajo será necesario, por tanto, profundizar un poco más en estos campos.

2.1. Inteligencia Artificial

2.1.1. Contexto histórico de la Inteligencia Artificial

Al hablar de Inteligencia Artificial, no es extraño pensar en algo novedoso, moderno, y, en definitiva, en una tecnología muy innovadora, pero bastante joven. Contrariamente a dicho pensamiento, la realidad es que el concepto de razonamiento artificial está presente desde muchísimo antes: ya en la mitología de la Grecia antigua se hace referencia a Talos, un gigante de cobre capaz de razonar por sí mismo, diseñado para proteger la isla de Creta [Hunter and Rhodius, 2015]. Sin embargo, y a pesar de las muchas referencias literarias y mitológicas, y a los varios intentos que se hicieron a partir del siglo XIV por formalizar el razonamiento artificial (como las "máquinas lógicas" de Ramón Llull [Bonner, 2007]), no es hasta mediados del siglo XX que se alcanzan una tecnología y una matemática lo suficientemente potente como para llevar a cabo este propósito.

Es a mediados de este siglo que Alan Turing, un importante matemático que será recordado, entre otras cosas, por sus enormes contribuciones a la computación, se planteó si es posible que una máquina llegara a razonar como un humano. A raíz de esto, publica en 1950 el artículo "Computing Machinery and Intelligence" [TURING, 1950], en el que desarrolla el conocido como Test de Turing: según él, una máquina es capaz de pensar,

y por tanto es inteligente, si es capaz de convencer a un humana de que ella es también humana. La inteligencia (o no) de una máquina se puede comprobar, por tanto, con lo que Turing denominó el juego de la imitación: en dos salas distintas se encuentran un humano y la máquina a examinar, sin que se sepa en qué sala está cada uno. Una persona deberá entablar conversaciones con ambas y, tras ello, deberá decir cuál (cree que) es el humano, y cuál es la máquina. Si una cantidad consistente de personas se equivoca en esta prueba, la máquina habrá sido capaz de imitar plenamente a un humano, y por tanto superar el test de Turing y ser considerada inteligente. En la actualidad, el test de Turing ya ha sido superado en varias ocasiones a través de chatbots (bots conversacionales), como ELIZA, A.L.I.C.E. [Wallace, 1995], o el mucho más reciente y potente ChatGPT [OpenAI, 2022]. Incluso, en 2022, un ingeniero de Google destinado al desarrollo del modelo de lenguaje LaMDA [Google, 2021] afirmó que este estaba vivo y que era una IA sintiente. [TheGuardian, 2022].

Tanto este ingeniero, como, en su momento, Alan Turing, no fueron capaces de convencer a todo el mundo. Para desacreditar el test de Turing, el filósofo John Searle publica en 1980 un artículo en el que describe lo que pasará a concerse como el argumento de la habitación china [Cole, 2020]. En este experimento mental, tenemos a una persona encerrada en una habitación con una única ranura mediante la cual puede recibir mensajes escritos, y responderlos. Los mensajes están escritos en chino y, aunque la persona encerrada no tiene ningún conocimiento de este idioma, posee una serie de manuales y diccionarios que le indican exactamente qué debe escribir en función del mensaje recibido. De esta manera, la persona que está fuera de la habitación siente que está comunicándose con un hablante experimentado de chino, cuando la realidad es bien distinta. Con este argumento, Searle pretende demostrar que el hecho de que una máquina se comporte como si fuera sintiente, no significa que lo sea. Aún hoy en día, filósofos de todo el mundo siguen trabajando para determinar si, efectivamente, una máquina puede ser inteligente y sintiente, y cómo determinarlo; un factor que, dados los últimos avances en este ámbito, resulta cada vez más apremiante.

2.1.2. Tipos de Inteligencia Artificial

Desarrollar un algoritmo que realice una determinada tarea puede resultar extremadamente complejo, o incluso pueden darse las circunstancias de que no exista el conocimiento necesario para formalizarlo. Un ejemplo muy común para este caso es el del reconocimiento de objetos. Algo tan "simpleçomo detectar mesas en una imagen es en realidad una ha-

zaña de gran envergadura, no solo por la complejidad de los algoritmos de visión artificial, que existen y obtienen un gran rendimiento, sino porque, para distinguir mesas de otros objetos, se debe tener muy claro qué es una mesa (y qué no lo es). Si consideráramos, por ejemplo, que una mesa es "un tablero horizontal con cuatro patas", nos daríamos cuenta de que esto incluye también bancos o taburetes, que no son mesas, mientras que excluye las mesas con una única pata central. Encontrar una definición formal que incluya a todos los elementos que son mesas, y excluya a todos los que no lo son, no resulta en absoluto trivial[1]. Las personas, sin embargo, son perfectamente capaces de distinguir qué es una mesa y qué no lo es. Esto se debe a que, mediante una larga serie de experiencias, y de ensayo/error, la persona ha aprendido qué es una mesa, sin necesidad de definirla explícitamente. [Caponera De Cobellis, 2021] Es esto mismo lo que se pretende conseguir mediante el Machine Learning.

El Machine Learning (ML) o Aprendizaje Automático, es una rama de la inteligencia artificial que busca desarrollar estrategias y algoritmos para que una máquina sea capaz de realizar tareas por sí sola, sin necesidad de ser programada explícitamente para ello, mediante técnicas de aprendizaje. Si bien la IA puede clasificarse según muchos criterios, como su funcionamiento o su capacidad, en esta sección se estudiará su clasificación según el método de aprendizaje. Algunos autores llegan a considerar hasta seis tipos de Machine Learning [Ayodele, 2010], sin embargo, en este documento se presentan tan solo los tres más importantes [Heidenreich, 2018].

Aprendizaje supervisado

Dentro de esta rama del Machine Learning se encuentran el Deep Learning y las redes neuronales, dos de los subtipos de IA que más éxito están teniendo en estos tiempos, y en los que se basan la mayoría de sistemas que intentan imitar el comportamiento humano. Es, precisamente, con aprendizaje supervisado, cómo suele resolverse el problema anteriormente mencionado de la identificación de objetos en una imagen. La idea detrás del aprendizaje supervisado es similar al uso de flashcards para el estudio: se suministra a la IA una serie de entradas (por ejemplo, fotografías) con su salida correspondiente (por ejemplo, la fotografía en cuestión contiene una mesa o no), y la labor de la IA será la

[1] A este respecto es interesante mencionar el dilema del sándwich y el perrito caliente (¿es el perrito caliente un sándwich?) [Carrizo Zirit, 2019]. A pesar de ser un tema de poca enjundia que se popularizó en redes sociales, refleja perfectamente el mismo problema que presenta la clasificación de mesas del ejemplo.

de aprender qué patrones relacionan las entradas con las salidas, para posteriormente ser capaz de extrapolar este conocimiento a datos nuevos (por ejemplo, fotos que no había visto durante su aprendizaje), tal y como se muestra en la Figura 2.1. Se debe tener mucho cuidado con el conjunto de datos que se le suministra a la IA, puesto que esto podría derivar en un *sobreentrenamiento* (overfitting) o *infraentrenada* (underfitting) [Na8, 2017]. El sobreentrenamiento se produce cuando el conjunto de datos no es lo suficientemente amplio o variado, de manera que la IA, en lugar de aprender los patrones subyacentes que relacionan las entradas con las salidas, simplemente "memoriza" los ejemplos, de modo que no reconocerá ningún dato fuera de los de entrenamiento. Por otro lado, si tan solo mostramos imágenes de mesas blancas durante el entrenamiento, la IA no será capaz de reconocer mesas de otro color[2]. Esto es similar a cuando un estudiante memoriza los problemas del libro sin entender por qué hace lo que hace: es muy probable que, cuando en el examen se le presente un problema distinto, no sea capaz de resolverlo.

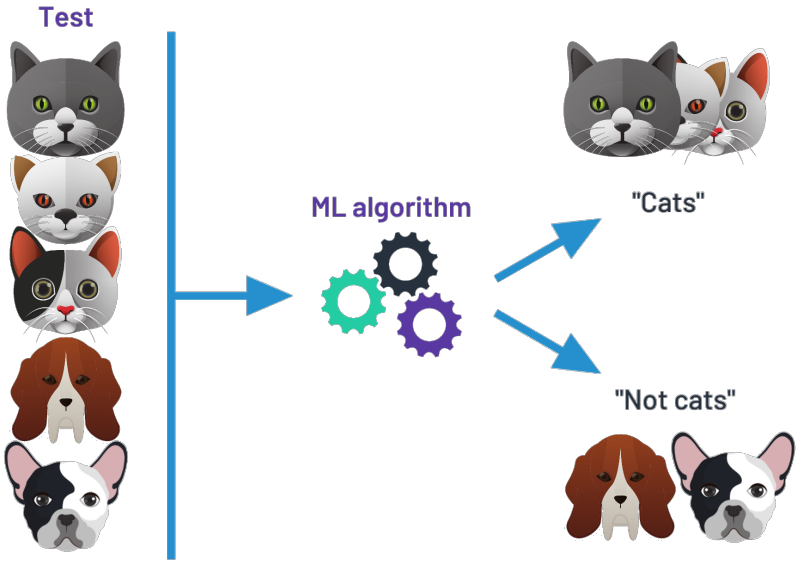


Figura 2.1: Representación de un algoritmo de aprendizaje supervisado entrenado para distinguir gatos de otros animales. Adaptado de [Pickel, 2021].

[2]Una llamativa demostración de esto puede verse en IAs como DeepNude, que fue entrenada exclusivamente con cuerpos femeninos, produciendo resultados completamente erróneos al utilizarse sobre hombres [Santana Vega, 2019].

Aprendizaje no supervisado

A pesar de sus muy buenos resultados, el aprendizaje supervisado cuenta con un problema no menor: es necesario contar con datos etiquetados en cantidades muy elevadas (mayores cuanto mayor sea la complejidad del problema), y para ello es necesario un proceso de etiquetado. Por ejemplo, para suministrarle a una IA la respuesta a si una foto determinada contiene una mesa, alguien debe examinar dicha foto y generar la respuesta primero.

Es por esto que existe otra familia de algoritmos que no requiere el etiquetado de los datos: el aprendizaje no supervisado. En este modelo de aprendizaje, se suministran una serie de características sobre los datos, y la IA buscará patrones entre ellos, y los clasificará. A diferencia del aprendizaje supervisado, una IA de este tipo, dada la foto de una manzana, jamás será capaz de decir que eso es una manzana, puesto que no concibe ese concepto. Sin embargo, será capaz de distinguir una manzana de una pera en función de las características suministradas (color, forma,...), y creará grupos separados para cada categoría, como se muestra en la Figura 2.2. Este tipo de algoritmos son muy utilizados en los recomendadores de contenido multimedia, como Netflix: a partir de las características de las películas que te han gustado, te recomendará otras similares. Otro uso que, en ocasiones se le da a este tipo de IA es el etiquetado de datos: se toman una serie de datos sin etiquetar, y se le proporcionan como entrada a una IA de aprendizaje no supervisado para que los "agrupe" según sus características. Posteriormente, un humano inspecciona cada uno de los grupos y los etiqueta (etiquetando, por tanto, todos los datos de ese grupo), obteniendo así un conjunto de datos etiquetados que pueden suministrarse a una IA de aprendizaje supervisado.

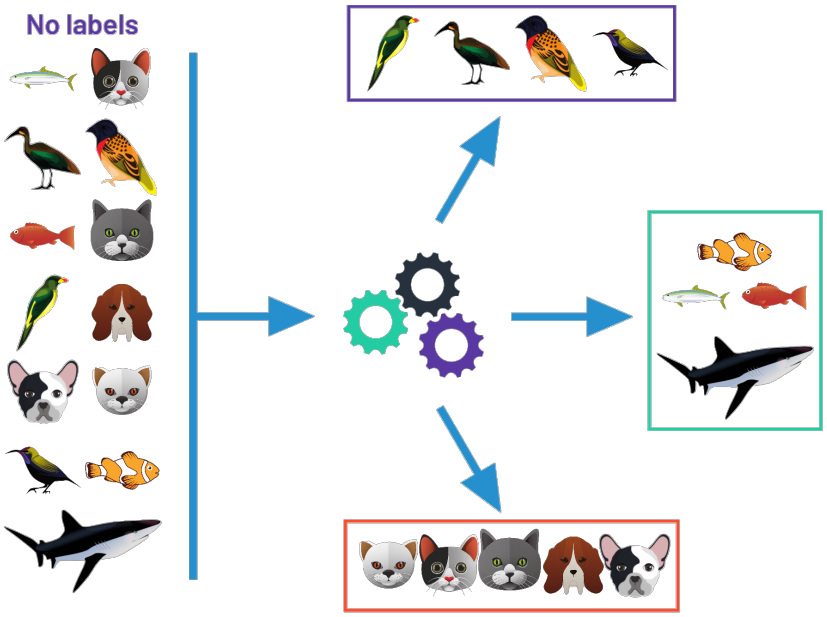


Figura 2.2: Representación de un algoritmo de aprendizaje no supervisado entrenado para agrupar animales en función de si son de tierra, agua o aire. Adaptado de [Pickel, 2021].

Aprendizaje por refuerzo

El aprendizaje por refuerzo cambia considerablemente el paradigma del aprendizaje respecto a los modelos desarrollados en secciones anteriores: con o sin etiquetas (salida esperada para una entrada dada), ambos tipos de aprendizaje utilizan un conjunto de datos, o dataset, para su entrenamiento. El aprendizaje por refuerzo, en cambio, aprende "sobre la marcha". La filosofía detrás de este tipo de aprendizaje es la de aprender de los errores. La IA, generalmente denominada *agente* en este contexto, aprende "probando" distintas acciones, observando el resultado y recibiendo una recompensa (o castigo) por ello. Su objetivo será el de maximizar la recompensa obtenida, priorizando las acciones que garanticen una recompensa alta, y evitando aquellas que generen un castigo. Este tipo de inteligencia artificial es especialmente utilizado en juegos y videojuegos, como es el caso de AlphaGo Zero [Silver et al., 2017], una IA desarrollada por Google Deepmind que fue capaz de derrotar a los mayores campeones del mundo en el juego del Go (considerado por muchos el juego de mesa más difícil, bastante por delante del ajedrez); pero también es ampliamente utilizado en el ámbito industrial y la robótica. Una de las grandes ventajas de este tipo de aprendizaje es que no requiere información sobre el entorno, ni una gran cantidad de datos de entrenamiento, sino que realiza el entrenamiento explorando el propio

entorno, lo cual lo hace ideal para problemas en los que no hay una cantidad suficiente de datos de calidad. La Figura 2.3 muestra un ejemplo de aprendizaje por refuerzo.

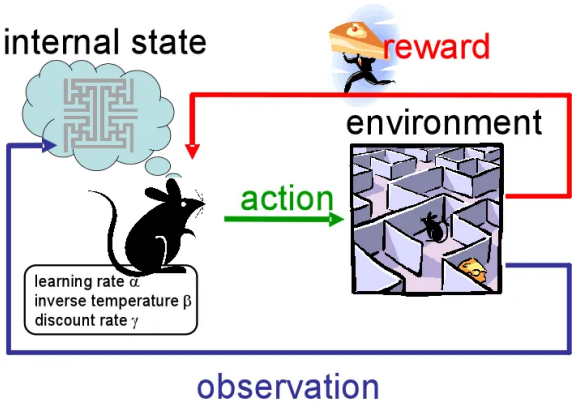


Figura 2.3: Representación de un algoritmo de aprendizaje por refuerzo. El ratón representa al agente, que debe moverse por el laberinto (entorno) para obtener la recompensa (comida). Extraído de [Das, 2017].

2.2. Computación Cuántica

2.2.1. Propiedades cuánticas de la materia

La mecánica cuántica es un modelo físico que surge a finales del siglo XIX y principios del siglo XX como una teoría que pretende explicar sucesos hasta la época incomprensibles que se habían comprobado experimentalmente a escala atómica. Al contrario de lo que se pensaba hasta entonces, la materia no se comporta a escalas atómicas igual que lo hace a escalas macroscópicas: un electrón, o un protón, tienen muy poco que ver con una canica con carga eléctrica, a pesar de que sea este el ejemplo más clásico para visualizar mentalmente dichas partículas. Algunos experimentos, como el de la doble rendija de Young [Young, 1804], contradecían las predicciones de los modelos teóricos vigentes, exhibiendo propiedades a veces contradictorias de la materia, como la dualidad onda-partícula, que establece que, según las circunstancias, una partícula se comportará como una onda o como un corpúsculo. Puesto que la mecánica cuántica es un campo de estudio muy amplio, y aún no del todo comprendido, se detallan a continuación, brevemente, las propiedades cuánticas de la materia que tienen una aplicación dentro del ámbito de la computación cuántica.

Principio de superposición

Esta es, como se verá más adelante, una de las propiedades más aplicadas en el contexto de la mecánica cuántica, y también una de las más anti-intuitivas, puesto que el cerebro humano, acostumbrado al mundo macroscópico, es incapaz de visualizar ciertos comportamientos cuánticos.

Lo que indica el principio de superposición es que una partícula, por ejemplo, un electrón, puede estar en varios estados "a la vez"[3] mientras no la estamos mirando. Esto no significa que, al mirar un electrón, este estará en dos lugares a la vez, sino que puede estar en cualquiera de los dos. Matemáticamente, se modela esto mediante una función de onda que mide la probabilidad de que el electrón se encuentre en uno u otro lugar. Cuando está en aislamiento, el electrón se encuentra en un estado de superposición, es decir, su posición está indeterminada[4]. Cuando se realiza una observación (es decir, cuando se interactúa con la partícula), se dice que la función de onda colapsa, y la partícula acaba determinada en una de las posibilidades descritas por su función de onda. Lo verdaderamente importante de la superposición es que no se trata de ningún error de medida, ni nada similar: si se toman dos sistemas exactamente iguales y se realizan las mismas acciones sobre ellos, es posible que se obtengan resultados distintos. Este fenómeno no ocurre únicamente con electrones, ni sobre la posición, sino que se aplica a una gran cantidad de propiedades de partículas subatómicas como el spin de un electrón, la polarización de un fotón, etc.

Los fotones suponen, tal vez, la manera más sencilla de visualizar este fenómeno a escala macroscópica, mediante el experimento del triple polarizador de Dirac [Caltech, 2023]. Para este experimento se utilizan, inicialmente, dos polarizadores de la luz: uno horizontal, y uno vertical. Puesto que un polarizador horizontal solo permite el paso a las ondas de luz horizontales, y uno vertical solo a las verticales, superponer ambos polarizadores implica que ninguna onda de luz pueda pasar (Figura 2.4).

[3] Estrictamente hablando, sería más correcto decir que la partícula no está en ninguno de esos estados, sino que está indeterminada, y al colapsar la función de onda se "decide" por una de las posibilidades.

[4] Es importante matizar que no se trata de que no seamos capaces de conocer la posición, y que por tanto sea desconocida, sino que está indeterminada: la partícula no tiene una posición, tan solo una determinada probabilidad de estar en uno u otro sitio.

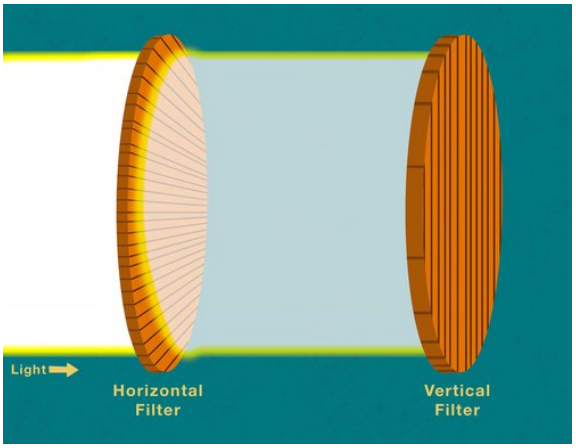


Figura 2.4: Experimento del triple polarizador de Dirac utilizando un polarizador horizontal y uno vertical. Extraído de [Caltech, 2023].

Sin embargo, si se añadiera un tercer polarizador orientado a 45° entre los dos, parte de la luz sí sería capaz de atravesar el conjunto, al contrario de lo que cabría esperar (Figura 2.5). Esto se debe a que, cuánticamente, puede entenderse un rayo polarizado diagonalmente como la superposición uniforme (equiprobable) de un rayo horizontal y otro vertical. Es decir, que un rayo polarizado 45º tiene un 50 % de probabilidades de "colapsar" a un rayo horizontal, y otro 50 % de colapsar a uno vertical. De igual modo, un rayo horizontal puede verse como la superposición uniforme de dos rayos orientados a 45° y -45°, y de manera análoga para un rayo vertical. Por lo tanto, lo que ocurre es que, cuando un rayo atraviesa el primer polarizador, tiene un 50 % de probabilidades de colapsar a un rayo inclinado 45º, atravesando también el segundo. A continuación, el rayo diagonal tiene un 50 % de probabilidades de colapsar a un rayo vertical, por lo que un 25 % de la luz total atraviesa el conjunto.

Esto resalta también un fenómeno muy importante, y es que la superposición depende de la base de medida: como se ha visto, un rayo horizontal puede expresarse como superposición de dos rayos diagonales perpendiculares. Por lo tanto, si para medir utilizáramos un polarizador diagonal, solo la mitad de los rayos lo atravesarían, por estar en superposición, puesto que el rayo horizontal tiene un 50 % de probabilidades de colapsar a un rayo inclinado 45º. Sin embargo, cuando el polarizador que utilizamos es horizontal, la superposición desaparece, puesto que un rayo horizontal tiene un 100 % de probabilidad de colapsar a un rayo horizontal. Por lo tanto, una partícula puede estar en superposición respecto de una base, y no estarlo respecto de otra.

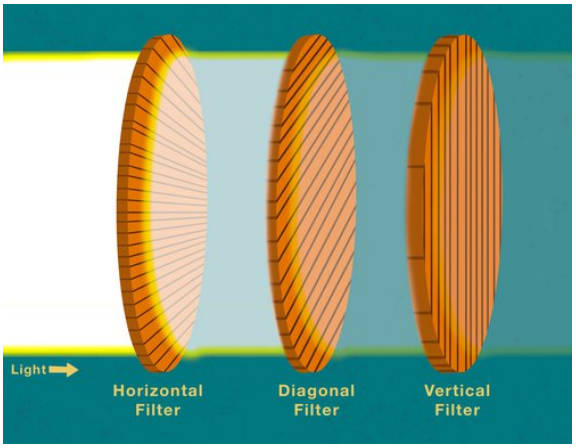


Figura 2.5: Experimento del triple polarizador de Dirac utilizando un polarizador intermedio inclinado a 45°. Extraído de [Caltech, 2023].

Efecto túnel

El efecto túnel es otro efecto que resulta muy anti-intuitivo, pero que se verifica experimentalmente. Este efecto es una consecuencia del principio de superposición anteriormente detallado.

En este caso, en lugar de tener una partícula aislada, se tiene una partícula que se intenta encerrar en una caja[5]. Si la partícula se comportara como una canica, esta tarea no tendría una gran dificultad. Sin embargo, se ha visto anteriormente que una partícula no tiene una posición bien definida, sino una función de probabilidades de estar en uno u otro lugar. En la Figura 2.6 se puede apreciar cómo, en el mundo cuántico, las paredes no impiden por completo el paso de las partículas, sino que únicamente disminuyen (enormemente) la función de probabilidad. Esto significa que, estadísticamente, llegará un momento en que la partícula encerrada se materializará fuera de la caja, y "escapará". Extrapolado al mundo macroscópico, sería como si, al lanzar un balón contra una pared un número inimaginablemente grande de veces, antes o después el balón atravesara la pared.

Este efecto también es utilizado en ciertos paradigmas de computación cuántica, y es uno de los motivos por los que la computación clásica se estancará pronto: los transistores se basan en "abrir y cerrar" un circuito, es decir, permitir o no el paso de la corriente. Con el avance tecnológico, se ha conseguido fabricar transistores de muy pocos nanómetros de tamaño pero, a estas escalas, el efecto túnel puede verificarse, haciendo que un electrón

[5]Estrictamente, se debe hablar de barreras de potencial, sin embargo, puesto que se pretende dar una idea intuitiva del fenómeno, resulta mucho más visual pensar en las paredes de una caja.

"salte" de un lado a otro del circuito aún cuando este debería estar "abierto" (no debería pasar corriente). Es por ello que cada vez resulta más complicado realizar transistores más pequeños, y será imposible continuar escalando.

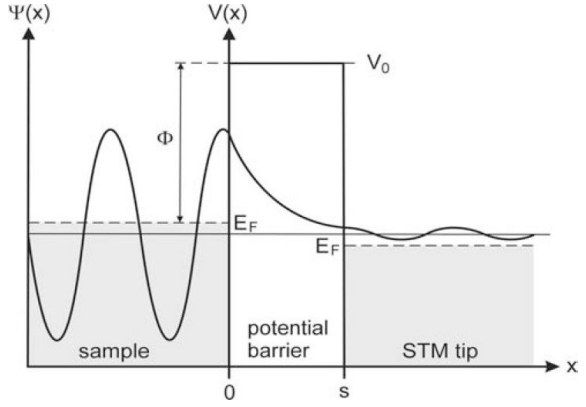


Figura 2.6: Visualización del efecto túnel. La barrera de potencial representa una de las paredes de la caja. Se aprecia cómo la función de onda es mucho más amplia en el interior de la caja (izquierda de la pared) que en el exterior, pero nunca se anula por completo. Extraído de [Trixler, 2013].

Entrelazamiento cuántico

Por último, el entrelazamiento cuántico es una propiedad que hace que dos partículas puedan entrelazarse y que, de ese modo, el estado de una partícula esté completamente ligado al estado de la otra. Por ejemplo, es posible entrelazar dos electrones de manera que tengan siempre el mismo spin. De este modo, cuando se interactúa con una de las dos partículas midiendo su spin, instantáneamente la otra partícula pierde su indeterminación y colapsa a tener el mismo spin que la primera, independientemente de la distancia a la que se encuentren. Este fenómeno, que Einstein calificó de "acción fantasmal a distancia" [Muller, 2022], parece contradecir las leyes de la física, en particular aquella que establece que nada puede viajar más rápido que la luz: ¿cómo pueden dos partículas "comunicarse" para cambiar su estado de manera instantánea sin importar la distancia? ¿No estaría la información viajando más rápido que la luz? La clave para responder estas cuestiones se halla en que las partículas, una vez se entrelazan, dejan de ser dos partículas independientes y pasan a ser parte de un único sistema, de manera que resulta imposible describir el estado de solo una de las partículas, sino que es necesario considerarlas en conjunto. De este modo, no es necesario que se "transmita" ninguna información entre las partículas, porque actúan

como una sola.

El entrelazamiento es la pieza final para comprender las propiedades físicas principales en las que se basa la computación cuántica.

2.2.2. Notación de Dirac o bra-ket

En apartados posteriores, se utilizará la denominada notación bra-ket o notación de Dirac, en honor a su creador Paul Dirac [Paolucci, 2020]. En esta sección se desarrolla, de modo superficial, el uso de los kets para representar qubits.

La finalidad de la notación de Dirac es simplificar la escritura de estados y operadores cuánticos. Principalmente, se compone de dos partes: el bra, y el ket. El ket, denotado como $|\rangle$, representa un estado cuántico. En el caso de la computación cuántica, el contenido del ket será, generalmente, una secuencia binaria que represente el estado de ejecución del computador o de una serie de qubits. De esta manera, $|0000\rangle$ representa cuatro qubits que están en el estado 0. Además, las propiedades de los ket establecen que es equivalente escribir $|01\rangle$ y $|0\rangle\otimes|1\rangle$, es decir, en ambos casos se representa la secuencia 01, sin embargo, puede ser más práctico utilizar una u otra notación según las circunstancias. La operación $\otimes$ representa el producto tensorial, que, generalmente, se omite, pudiendo escribirse lo anterior como $|0\rangle|1\rangle$. Es habitual utilizar este tipo de notación cuando el estado está "compuesto" por varias partes, por ejemplo, una entrada y una salida, dos entradas y una salida, etc. Esto puede denotarse como

$$|e\rangle|s\rangle = |e, s\rangle$$

o

$$|e_1\rangle|e_2\rangle|s\rangle = |e_1, e_2, s\rangle,$$

respectivamente, donde e representa las entradas y s la salida. Notar cómo en ambos casos se refiere a entradas o salidas genéricas, sin hacer referencia al número de qubits de cada uno, permitiendo generalizar la notación.

Utilizando esta notación es posible representar la superposición de estados como una suma de los distintos estados superpuestos, ponderados por su probabilidad, de manera que se representa un estado ψ como

$$|\psi\rangle = \sum_i \alpha_i |i\rangle,$$

donde la probabilidad de un estado $|i\rangle$ se calcula como α_i^2. Por supuesto, al tratarse de una probabilidad, la suma de todas las probabilidades debe ser exactamente igual a 1, es decir,

$$\sum_i \alpha_i^2 = 1.$$

De este modo, se puede representar la superposición uniforme de un qubit como

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

donde tanto el valor 0 como el valor 1 tienen una probabilidad de

$$\left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2} = 50\,\%$$

de aparición. Además, el producto tensorial de kets cumple la propiedad distributiva, de modo que

$$|0\rangle\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle).$$

Así, la superposición uniforme de dos qubits se representaría como

$$\begin{aligned}\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)^{\otimes 2} &= \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) \\ &= \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle.\end{aligned}$$

A los estados

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

y

$$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

se los denota, respectivamente, por $|+\rangle$ y $|-\rangle$.

2.2.3. Modelos de computación cuántica

Una vez se conocen las principales características que distinguen a la computación cuántica de la clásica, es necesario comprender cómo estas propiedades afectan a la computación, y cómo es posible aprovecharlas para obtener una tecnología más eficiente. Principalmente existen dos tipos de computadores cuánticos: los que están basados en puertas, que son universales; y los que se basan en annealing, que están orientados a la optimización.

Computadores universales basados en puertas cuánticas

Los computadores cuánticos universales son los que permiten realizar cualquier operación o que permiten resolver problemas de varios dominios, ya sea optimización, factorización en números primos, búsquedas, etc.

La principal ventaja que ofrecen los computadores cuánticos es el denominado *paralelismo cuántico*. Esto consiste en aprovechar la propiedad de superposición cuántica para representar, simultáneamente, varios estados. Si, en computación clásica, una secuencia de n bits puede representar un único valor, una secuencia de qubits de la misma longitud puede representar 2^n valores simultáneamente, es decir, todas las combinaciones posibles. Esto se debe a que cada qubit puede encontrarse en una superposición de 0 y 1. De este modo, si se diseña un circuito que sume 1 al valor de entrada, aprovechando la superposición es posible calcular, paralelamente, la suma a todos los posibles valores de entrada. Por ejemplo, supóngase una entrada de 2 qubits en superposición uniforme

$$|{+}{+}\rangle = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right),$$

y una salida de tres qubits (necesarios para representar el valor máximo posible, 4), que inicialmente vale $|000\rangle$, de modo que el estado inicial del algoritmo es $|{+}{+}\rangle\,|000\rangle$[6,7]. Es aquí donde cobra importancia el entrelazamiento: se deben asociar las entradas con las salidas correspondientes, porque de lo contrario lo único que se obtendrá será una superposición uniforme en todos los qubits. Es decir, se debe procurar que siempre que la entrada valga $|e\rangle$, la salida valga $|e+1\rangle$, esto es, obtener estados de la forma $|e\rangle\,|e+1\rangle$. La manera de lograr esto es mediante el entrelazamiento, entrelazando los qubits de entrada y los de salida para que no pueda existir ninguna combinación fuera de las deseadas $|e\rangle\,|e+1\rangle$.

Tras esto, sin embargo, el estado es una superposición uniforme de todas estas combinaciones, es decir,

$$\frac{1}{2}\sum_{e=0}^{3}|e\rangle\,|e+1\rangle = \frac{1}{2}(|00\rangle\,|001\rangle + |01\rangle\,|010\rangle + |10\rangle\,|011\rangle + |11\rangle\,|100\rangle).$$

[6]Nótese cómo se utilizan dos kets distintos para facilitar la distinción entre los qubits de entrada y los qubits de salida.

[7]A diferencia de los algoritmos clásicos, donde se puede alterar un bit de la entrada para producir la salida, los algoritmos cuánticos utilizan, frecuentemente, qubits distintos para la entrada y la salida. Esto se debe a que las operaciones deben ser, por definición, reversibles. Aunque, en ocasiones, es posible reutilizar los qubits de la entrada, en este caso se utilizan qubits distintos para la salida con el fin de ilustrar mejor el ejemplo.

Esto significa que cualquiera de los estados posibles tiene la misma probabilidad de aparecer[8], aunque solo se tenga interés en uno de ellos, por ejemplo en la salida asociada al valor $|01\rangle$. Por supuesto, este es un ejemplo sencillo, y carece de sentido utilizar paralelismo cuántico para calcular la suma 1+1. Sin embargo, en algoritmos más complejos, sí es importante obtener un valor en particular, en lugar de que haya una superposición uniforme. Es por esto que buena parte de los algoritmos cuánticos se centran en modificar las probabilidades de los distintos estados para maximizar la del estado deseado de manera que, en la mayoría de los casos, el resultado obtenido sea satisfactorio.

Basados en este tipo de tecnología, existen varios algoritmos que permiten

Quantum annealing

A diferencia de los computadores universales, los computadores basados en quantum annealing se utilizan para un propósito específico, el de optimización, siendo muy eficientes en este ámbito, pero no pudiendo resolver eficientemente otro tipo de problemas[9].

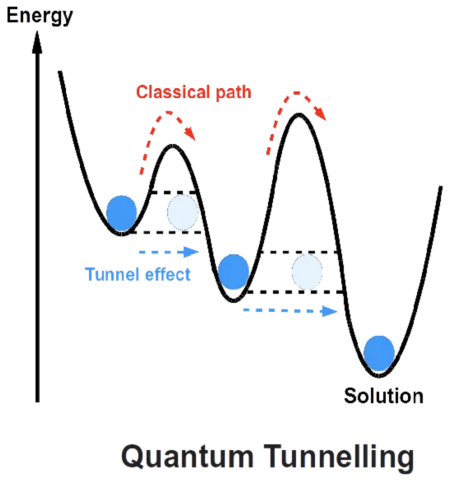
El simulated annealing, o templado simulado, es un método de optimización que ya se utiliza en computación clásica, y que se basa en el proceso de templado de los metales. El templado de metales permite aumentar la calidad final del producto, que depende en gran medida de la estructura que forman sus átomos. Para ello, se calienta el metal a gran temperatura, y posteriormente se deja enfriar bajo condiciones controladas, haciendo que los átomos, de manera natural, se organicen de la forma más óptima, conformando un reticulado altamente estructurado, y minimizando la energía del sistema. Este mismo proceso se ha intentado simular computacionalmente como método de optimización, con el fin de evitar los mínimos locales de la función.[Millán Páramo et al., 2014]

Una versión cuántica de este algoritmo, el quantum annealing, consiste en implementar físicamente la función que se desea minimizar, es decir, diseñar un sistema físico cuyos niveles de energía coincidan con la función a optimizar. Puesto que, de manera natural, todos lo sistemas tienden siempre a un estado de mínima energía, este algoritmo permite encontrar el mínimo de la función midiendo el estado en el que el sistema se estabiliza.

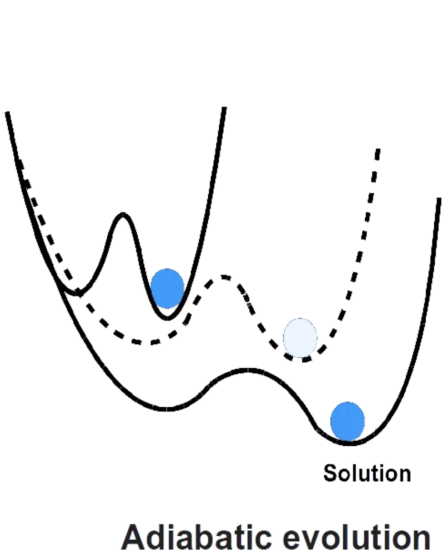
[8]Téngase en cuenta que, aunque los cuatro estados coexisten simultáneamente, solo es posible medir uno de ellos.

[9]En realidad, es posible utilizar quantum annealing para resolver problemas que normalmente se resolverían con puertas lógicas, sin embargo, el coste aumenta y la eficiencia disminuye enormemente, por lo que no resulta interesante [Cervera-Lierta, 2018].

El quantum annealing ofrece una ventaja respecto a los métodos clásicos, debida a una de las propiedades enunciadas anteriormente: el efecto túnel. Este efecto permite que, si se dan las condiciones, una partícula realice un salto de energía "fuera" del mínimo local, atravesando la barrera de potencial, y alcanzando un estado de menor energía, tal y como se muestra en la Figura 2.7a. De este modo, las posibilidades de hallar el mínimo absoluto de la función se incrementan enormemente.



(a) Representación del proceso de quantum annealing. El sistema (círculo azul) se mueve por la función aprovechando el efecto túnel para "atravesar" las barreras de energía que limitan el mínimo local, permitiendo alcanzar un mínimo absoluto. En rojo se muestra el recorrido que debe realizar un simulador de annealing clásico, que resulta más costoso computacionalmente.



(b) Representación del proceso de evolución de un computador adiabático. Se parte de una función bien conocida, y lentamente se transiciona a la función objetivo, garantizándose que, una vez la transición esté completa, se hallará el mínimo absoluto.

Figura 2.7: Representación de quantum annealing basado en el efecto túnel (2.7a) y en computación adiabática (2.7b). Adaptado de [Cervera-Lierta, 2018].

Sin embargo, no es sencillo saber si, efectivamente, el valor que se alcanza es el mínimo absoluto, o tan solo un mínimo local con un nivel de energía muy bajo. Dependiendo del problema, puede ser necesario asegurarse de que, efectivamente, se trata de un mínimo absoluto. Esto es posible utilizando otro método cuántico, englobado también dentro del quantum annealing, denominado computación adiabática. Este proceso consiste en preparar inicialmente el sistema con una función de energía bien conocida, sencilla de im-

plementar, y de la que se conoce su mínimo, situando ahí el sistema. A continuación, de manera adiabática (es decir, muy, muy lentamente), se hace transicionar esta función de energía hacia la que se desea optimizar, realizando pequeñas variaciones y recalculando el mínimo de la nueva función iterativamente, hasta alcanzar la función deseada. Si el proceso se realiza lo suficientemente lento, el teorema adiabático garantiza que el estado de mínima energía que alcanza el sistema es la solución óptima del problema [Cervera-Lierta, 2018].

3. Objetivos

El objetivo principal de este trabajo es el estudio de la viabilidad del uso de tecnologías de computación cuántica en el ámbito del aprendizaje por refuerzo. Para ello, se ha seleccionado el algoritmo Q-Learning, por ser uno de los más utilizados y con mejores resultado en este campo. Se busca desarrollar una política de selección de acciones para Q-Learning basada en estas tecnologías, para posteriormente comparar su rendimiento con tres variantes clásicas del algoritmo: ϵ-greedy, SoftMax y Upper Confidence Bound. Para ello, se utilizará el software disponible en [Caponera De Cobellis, 2021], que permite probar distintas políticas de Q-Learning en el entorno FrozenLake de OpenAI Gym. Después de estudiar las distintas políticas, se hará un análisis comparativo de todas las variantes.

Además, es un objetivo de este trabajo el ser comprensible para públicos no especialistas dentro del ámbito del Machine Learning o de la computación cuántica, limitando para ello las formulaciones matemáticas y la terminología específica a lo imprescindible, e intentando crear intuición sobre las afirmaciones que se realizan siempre que sea posible, manteniendo siempre un nivel de rigor acorde a un trabajo académico como el presente.

4. Desarrollo del trabajo

4.1. Aprendizaje por refuerzo

Si bien en la sección 2.1.2 ya se ha dado una idea intuitiva y superficial del aprendizaje por refuerzo, en esta sección se aborda el tema con más detalle. El aprendizaje por refuerzo, como se ha dicho, consiste en "aprender de los errores". Más concretamente, este proceso de aprendizaje puede dividirse en seis fases [Shyalika, 2019]:

- **Observación del entorno.** El agente interactúa con el entorno obteniendo información, lo que se concreta en el estado del agente respecto al mismo

- **Toma de decisión.** El agente debe elegir una de las acciones posibles a partir de su estado actual. Este proceso de decisión se hace siguiendo una *política de selección de acciones*, que no es más que una estrategia que determina qué acción es más conveniente. Hallar esta estrategia no es trivial, y las distintas variantes del algoritmo surgen precisamente de utilizar políticas diferentes. Como se detallará más adelante, este trabajo se centra en la aplicación de la computación cuántica en la selección de acciones.

- **Ejecución de la acción.** Una vez que el agente, mediante la política, ha establecido cuál es la mejor acción, debe ejecutarla. Esto causará cambios en el entorno y en el estado del agente.

- **Recompensa.** Para que el agente sea capaz de aprender, debe ser recompensado (o castigado) en función de sus acciones. Tras ejecutar una acción, el entorno asignará una recompensa (que puede ser nula), que ayudará al agente a aprender.

- **Mejora de la estrategia.** Al recibir una recompensa, el agente podrá determinar la calidad de la acción tomada, y en función de eso deberá modificar su política: una acción que reporta una recompensa alta debe ser "reforzada", mientras que una acción que reporta un castigo debe intentar evitarse. Es en este punto donde se puede decir que el agente está aprendiendo.

- **Repetición.** Tras los cinco pasos anteriores, el agente ha sido capaz de mejorar levemente su estrategia. Es necesario repetir este proceso una cantidad suficiente de

veces para que el agente obtenga el conocimiento suficiente como para ser capaz de resolver el problema para el que está siendo entrenado. El entrenamiento finalizará una vez que la estrategia del agente es lo suficientemente buena. Cómo determinar cuándo la estrategia es lo suficientemente buena, sin embargo, dependerá de cada problema en específico.

Los algoritmos de aprendizaje por refuerzo pueden ser clasificados, principalmente, según dos criterios:

- **Según su uso de un modelo**: los algoritmos *model-based* son aquellos que tienen cierto conocimiento sobre el funcionamiento del entorno y poseen un modelo interno sobre cómo este asigna las recompensas y la forma de la función de transición[1], y utiliza esta información para estimar la política óptima. Los algoritmos *model-free*, por el contrario, no utilizan ningún modelo interno, y se basan únicamente en las recompensas obtenidas para estimar la política. Q-Learning es un algoritmo *model-free*.

- **Según la política que calcula**. En este caso, es importante distinguir entre la política que el agente utiliza durante el entrenamiento, y la política que calcula como resultado: el resultado del entrenamiento será una política π que, dado un estado, devuelve la acción más conveniente, pero es estática, es decir, no cambia al recibir la recompensa; por el contrario, el agente utiliza una política durante el entrenamiento para decidir qué acción tomar, pero esta política no tiene por qué ser la misma que se genera como resultado del entrenamiento. Se llama, por tanto, *on-policy* a los algoritmos que utilizan durante el entrenamiento la misma política que están calculando, mientras que son *off-policy* aquellos que utilizan políticas distintas, como es el caso de Q-Learning.

4.2. Q-Learning

Q-Learning es uno de los muchos algoritmos que pertenecen a la familia del aprendizaje por refuerzo. Toma su nombre de la matriz Quality (generalmente llamada simplemente matriz Q) que utiliza para la estimación de la política. Esta matriz tiene tantas filas como estados e_i tenga el entorno, y tantas columnas como acciones a_i (o al revés), y en

[1] Aquella función que determina, en función del estado actual y la acción tomada, el estado siguiente.

cada uno de sus elementos se guarda el *valor esperado* de tomar dicha acción desde el estado en cuestión. El valor esperado de una combinación de estado y acción (e_i, a_j) es, esencialmente, la recompensa que se estima que se obtendrá tomando la acción a_j desde el estado e_i. Por lo tanto, la matriz Q mide la bondad de tomar cada una de las acciones desde cada uno de los estados, de modo que cuanto mejor sea la recompensa esperada mejor será la acción[2]. En la Tabla 4.1 puede verse una representación de la matriz Q.

Q	a_1	a_2	a_3	a_4
e_1				
e_2				
e_3				
e_4				

Tabla 4.1: Representación de la matriz Q de un entorno con 4 estados y 4 acciones. No se muestran los valores esperados.

De la existencia de esta matriz puede inferirse que Q-Learning es un algoritmo diseñado para problemas con un número finito de acciones y de estados discretos, puesto que sería imposible construir la tabla en otro caso. Existen, sin embargo, modificaciones del algoritmo que permiten utilizar espacios de estados continuos [Barty et al., 2007], aunque no se estudian en este trabajo.

Los valores de la matriz Q se inicializan a un valor arbitario, que por lo general será el 0, y a continuación se inicia el proceso de entrenamiento, siguiendo los seis pasos descritos en la sección 4.1. El aprendizaje, es decir, la actualización de los valores de la matriz Q, se realiza utilizando la siguiente ecuación de Bellman, también conocida como ecuación del Q-Learning:

$$Q_{t+1}(e, a) = Q_t(e, a) + \alpha[R(e, a) + \gamma \max_{a'_i} Q_t(e', a'_i) - Q_t(e, a)] \tag{4.1}$$

donde:

- e representa el estado actual y a la acción tomada
- e' representa el estado futuro y a'_i las acciones disponibles desde e'

[2]Nótese que la acción no tiene por qué ser la mejor, sin embargo, a partir de la información de la que dispone el agente, es la que aparentemente reportará la mayor recompensa.

- $Q_t(e, a)$ representa el valor de la matriz Q en el tiempo t para el estado e y la acción a
- $R(e, a)$ representa la recompensa obtenida al ejecutar la acción a desde el estado e
- α representa el primer hiperparámetro del modelo: la tasa de aprendizaje o learning rate
- γ representa otro hiperparámetro: la tasa de descuento o discount rate

Como puede verse, la ecuación contiene dos hiperparámetros. Los hiperparámetros son parámetros del modelo que regulan su aprendizaje, y que por tanto deben ser elegidos por el desarrollador, en lugar de ser aprendidos por el modelo. Para comprender el funcionamiento del algoritmo, resulta imprescindible construir cierta intuición sobre la ecuación del Q-Learning, y sobre el significado de sus hiperparámetros, que se detalla a continuación.

Hiperparámetros del modelo: tasa de aprendizaje (α)

Analizando la ecuación (4.1), puede verse que la tasa de aprendizaje α es, esencialmente, un factor multiplicador de la variación del valor de Q. Esto puede verse más claramente reescribiendo la fórmula como

$$Q_{t+1}(e, a) = Q_t(e, a) + \alpha \left(\Delta Q_t(e, a)\right)$$

donde

$$\Delta Q_t(e, a) = R(e, a) + \gamma \max_{a'_i} Q_t(e', a'_i) - Q_t(e, a). \tag{4.2}$$

De este modo, se puede deducir que el la tasa de aprendizaje representa la magnitud del cambio de Q en cada iteración, es decir, un valor alto de α hará que se produzcan variaciones grandes en los valores de Q, mientras que valores bajos harán que el entrenamiento evolucione más lentamente. Por ello, debe tomar siempre valores reales en $[0, 1]$.

Encontrar un valor de α adecuado no es trivial, y requiere de conocimiento experto o, incluso, puede llevarse a cabo un proceso de optimización utilizando otros algoritmos de machine learning para ello. Al contrario de lo que se podría pensar, un valor alto de α no implica necesariamente un aprendizaje más rápido. Se puede pensar en la tasa de aprendizaje como en el tamaño del "paso" que da el agente. Si el tamaño del paso es demasiado grande, es posible que jamás alcance el punto óptimo y oscile indefinidamente alrededor del mismo, puesto que el paso es más grande que la distancia que lo separa

del objetivo. Esto puede verse representado en la Figura 4.1[3], en la que se muestra la función de recompensa (la recompensa que el agente obtiene en función de $Q(e, a)$), y las recompensas obtenidas por el agente. Es importante destacar que, aunque en este caso se está representando, el agente **no** conoce la función de recompensa. Además, se debe tener en cuenta que esta función se ha dibujado como una parábola para facilitar su comprensión, sin embargo la función real de recompensa será, probablemente, más compleja.

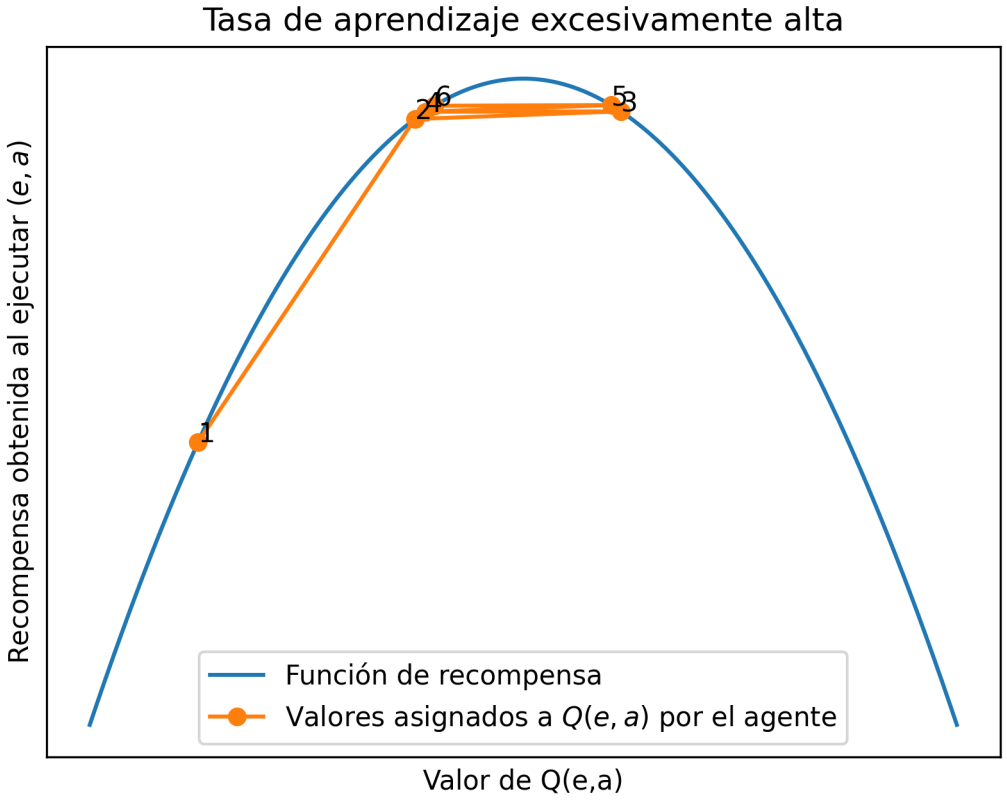


Figura 4.1: Representación de la recompensa obtenida con una tasa de aprendizaje demasiado alta. Se puede ver cómo, a partir de la segunda iteración, el agente empieza a oscilar alrededor del máximo, sin alcanzarlo.

También podría pensarse que, con el fin de evitar esta oscilación del agente alrededor del máximo, es conveniente utilizar una tasa de aprendizaje baja. Esto, si bien es cierto, conlleva un aprendizaje muy lento, puesto que son necesarias muchas más iteraciones para alcanzar el objetivo, tal y como se muestra en la Figura 4.2.

Por tanto, debe buscarse una tasa de entrenamiento óptima, que permita alcanzar el

[3]En esta y posteriores figuras, el agente puede "moverse" por el eje de abscisas, mientras que el eje de ordenadas representa la recompensa que el entorno le devuelve. Las distintas iteraciones están numeradas para facilitar el seguimiento de la evolución de los valores.

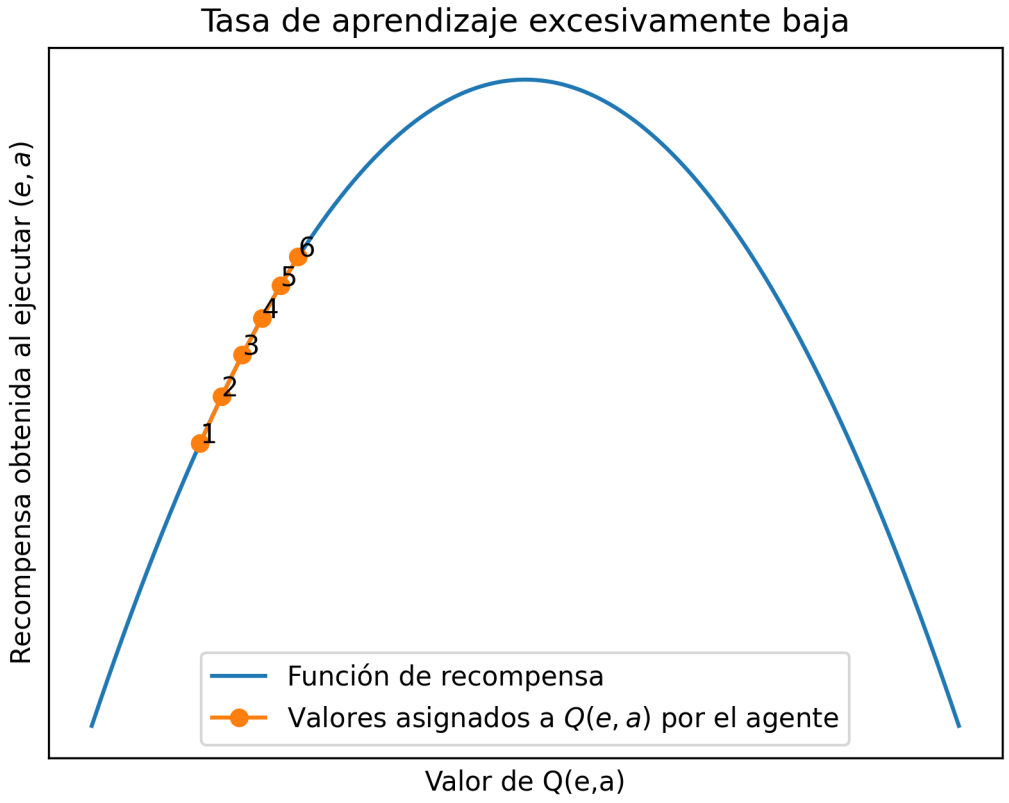


Figura 4.2: Representación de la recompensa obtenida con una tasa de aprendizaje demasiado baja. Se puede ver cómo el agente se acerca muy lentamente al punto máximo, necesitando un gran número de iteraciones para completar el entrenamiento.

objetivo en un número de pasos razonable, tal y como se muestra en la Figura 4.3.

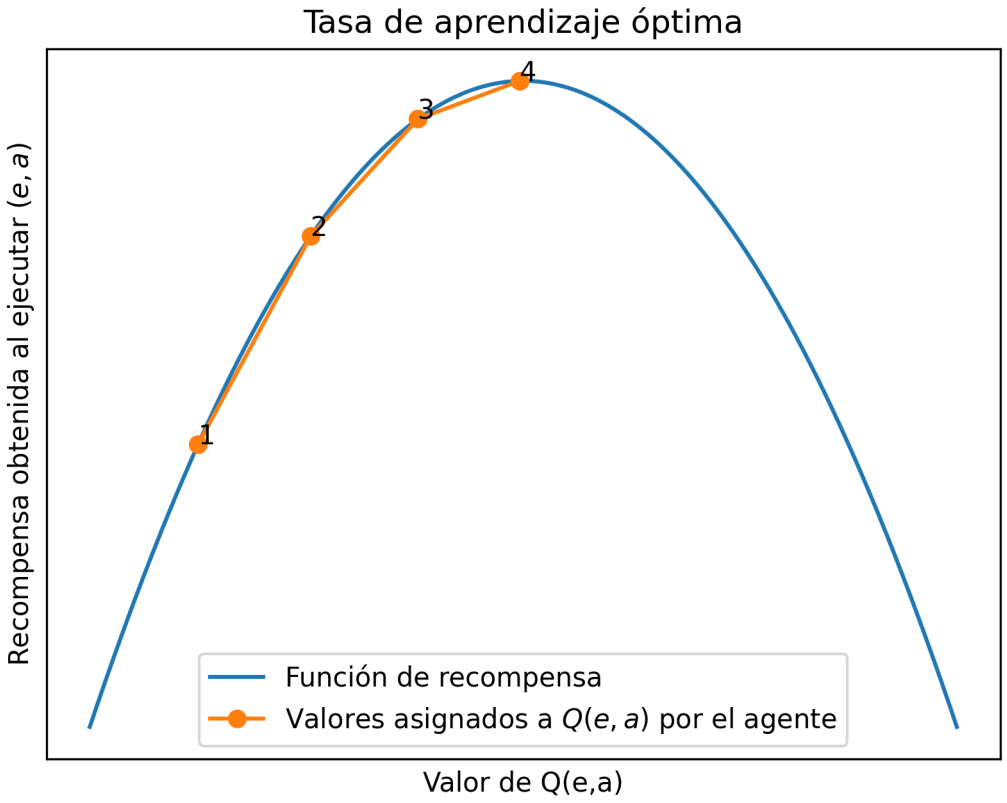


Figura 4.3: Representación de la recompensa obtenida con una tasa de aprendizaje óptima. Se puede ver cómo el agente alcanza el máximo en pocos pasos.

Hiperparámetros del modelo: tasa de descuento (γ)

Analizando el factor definido en la ecuación (4.2) como $\Delta Q_t(e,a)$, que representa la variación en el valor de Q, y que se reescribe en (4.3) para facilitar la lectura, se observan tres sumandos.

$$\Delta Q_t(e,a) = R(e,a) + \gamma \max_{a'_i} Q_t(e',a'_i) - Q_t(e,a). \tag{4.3}$$

Considerando únicamente el primero y el último,

$$R(e,a) - Q_t(e,a),$$

puede interpretarse como la diferencia entre la recompensa que el agente esperaba obtener $Q_t(e,a)$ y la que finalmente ha obtenido $R(e,a)$, siendo, en cierto modo, un factor de corrección del valor esperado.

El segundo término, $\gamma \max_{a'_i} Q_t(e',a'_i)$, aplica el multiplicador γ a $\max_{a'_i} Q_t(e',a'_i)$, que representa el máximo valor esperado para el estado al que se llega tras ejecutar la acción.

Es decir, $\max_{a'_i} Q_t(e', a'_i)$ mide cómo de bueno es el estado e', asumiendo que se tome la mejor acción posible. De este modo, se tiene en cuenta no solo la recompensa actual obtenida, sino también la expectativa de recompensa futura. El factor γ, por tanto, regula qué tanta importancia se le da a la recompensa futura en relación con la recompensa actual: con un $\gamma = 0$, se ignora por completo la recompensa futura, convirtiéndose en un algoritmo *greedy* (voraz), mientras que un $\gamma = 1$ se centrará en la recompensa futura más que en la actual. Al tener en cuenta recompensas futuras, los estados cercanos a otros estados con un gran valor esperado aumentarán también su valor esperado, haciendo que las recompensas obtenidas se "propaguen". La tasa de descuento γ mide con qué velocidad se propagan las recompensas.

Cabría pensar, una vez más, que, puesto que el objetivo del agente es maximizar la recompensa total, resulta interesante mantener una tasa de descuento alta, puesto que hará que el algoritmo se centre en la recompensa final y no en la actual. Esto, sin embargo, no es necesariamente así porque, según el entorno, las acciones tienen repercusiones de duración variable, y puede que no tenga sentido considerar efectos de estados futuros tan distantes, lo que generaría "ruido" introduciendo información innecesaria en la política.

Para visualizar mejor este concepto, supóngase la siguiente situación [clwainwright, 2019]: Supóngase una persona que, el primer día de cada mes, toma un helado y, para ello, debe decidir cuál le gusta más. Inspirándose en el aprendizaje por refuerzo, decide que juzgará la calidad del helado en función de las recompensas (por ejemplo, anímicas) que este proporcione. Estas recompensas pueden incluir qué tanto le gusta el sabor del helado al tomarlo (recompensa inmediata), pero también puede considerar si, por ejemplo, le sienta mal unas horas después de tomarlo. Del mismo modo, podría considerar si le sienta mal al día siguiente, a los dos días, o tantos días como quisiera. Sin embargo, si, por ejemplo, a las dos semanas se encuentra mal por haber tomado comida en mal estado, no sería lógico que esto repercuta negativamente en la evaluación del helado, y esto es porque el efecto del helado no es tan duradero. El número de días que considere representa, en cierto modo, el factor de descuento, es decir, cuánta importancia se le da a las recompensas futuras. Por tanto, del mismo modo, no debe elegirse una tasa de descuento demasiado alta, para evitar influenciar una decisión según factores ajenos a ella.

El algoritmo de Q-Learning puede formalizarse como se muestra en el Algoritmo 1.

Algoritmo 1: Q-Learning

$Q \leftarrow$ inicializar arbitrariamente;
for *episodio* **do**
  $e \leftarrow$ estado inicial;
  while *e no es un estado final* **do**
    $a \leftarrow$ acción escogida según la política de selección de acciones;
    Ejecutar la acción a;
    $r \leftarrow$ recompensa observada;
    $e' \leftarrow$ nuevo estado observado;
    $Q(e,a) \leftarrow Q(e,a) + \alpha[R(e,a) + \gamma \max_{a'_i} Q(e', a'_i) - Q(e,a)]$;
    $e \leftarrow e'$
  end
end

Llegados a este punto, y vista la complejidad del algoritmo, cabe preguntarse por qué debería utilizarse, en lugar de elegir siempre la acción con mayor valor esperado. En primer lugar, esto no es posible porque el agente no tiene conocimiento de su entorno, por lo que no conoce los valores esperados de las distintas acciones. Aún así, cabría plantearse la posibilidad de realizar una exploración en la que se probaran todas las acciones desde todos los estados una única vez y, a partir de esta información, elegir siempre el valor más alto, o incluso utilizar algoritmos de búsqueda en espacio de estados como A* [Hart et al., 1968] que tienen un reconocido alto rendimiento. La respuesta es que, a diferencia de estos algoritmos, Q-Learning está diseñado para entornos *no-deterministas*, esto es, entornos en los que ejecutar dos veces la misma acción desde el mismo estado puede producir resultados diferentes. En consecuencia, no es posible seguir la dinámica de algoritmos como A* que primero trazan un plan y después lo ejecutan, puesto que al ejecutar una acción, probablemente, el resultado no será el planificado, y el plan trazado deberá ser desechado.

4.2.1. Procesos de Decisión de Markov y FrozenLake

Para resolver el problema del no-determinismo en el caso de Q-Learning, los problemas se modelan como Procesos de Decisión de Markov (MDP) [Rocca, 2019]. Los MDP forman un marco matemático para representar decisiones donde no se tiene seguridad sobre el

resultado de las mismas, siempre y cuando se cumpla la propiedad de Markov.

Esta propiedad, que puede verse esquematizada en la Ecuación (4.4), esencialmente exige que las probabilidades de los distintos eventos dependan únicamente de la situación actual, y no de estados pasados, es decir, que la probabilidad de acabar en un determinado estado e' dependa únicamente del estado actual e y de la acción a ejecutar a, pero sea independiente del camino que se haya seguido para llegar a e.

$$P(\text{futuro}|\text{presente}, \text{pasado}) = P(\text{futuro}|\text{presente}) \tag{4.4}$$

Es esto lo que permite que, en cada paso, Q-Learning únicamente deba ser consciente de su estado actual y de las posibles acciones y, aunque pueda parecer un conjunto de problemas muy concreto, lo cierto es que pueden modelarse una enorme cantidad de situaciones como Procesos de Decisión de Markov, e incluso pueden adaptarse algunos problemas que, a priori, podrían parecer no cumplir la propiedad de Markov, de manera que sea posible utilizar las técnicas de resolución conocidas para los MDP.

El problema que se utiliza como campo de pruebas en este trabajo, FrozenLake [OpenAI, 2021], consiste en un entorno no determinista representado por un tablero 4x4 u 8x8, en el que las casillas pueden ser de cuatro tipos:

- Casilla inicial, en la que el agente se encuentra al principio de cada episodio
- Casilla objetivo, aquella a la que el agente debe llegar. Es una casilla final, es decir, al alcanzarla el episodio finaliza, y se recibe una recompensa de 1.
- Casillas congeladas, sobre las que el agente puede caminar. Estas casillas son resbaladizas, por lo que el agente tiene una probabilidad del 20 % de moverse en una dirección perpendicular a la seleccionada. Por ejemplo, si ejecuta la acción de moverse a la derecha, tendrá un 80 % de probabilidades de alcanzar la casilla de su derecha, un 10 % de alcanzar la casilla superior, y un 10 % de alcanzar la inferior[4]. Esto es lo que hace que el problema deba resolverse utilizando aprendizaje por refuerzo en lugar de búsqueda de caminos. Según la versión, pasar por estas casillas puede no

[4]Estas probabilidades son relativas a la versión 0.9.6, que es la que se utiliza en este trabajo. Desde que OpenAI Gym se convirtió en Farama Gymnasium en 2022, las probabilidades son del 33 % para cada dirección.

ofrecer ninguna recompensa, u ofrecer una pequeña penalización, con el fin de que el agente recorra el menor número de casillas posible.

- Por último, las casillas con agujeros, por las que el agente no puede pasar y, de intentarlo, se congelaría. También son estados finales y, según la versión, pueden no ofrecer ninguna recompensa, u ofrecer un castigo de -1.

Una representación del entorno FrozenLake puede verse en la Figura 4.4

Figura 4.4: Representación pixel-art del entorno FrozenLake. Extraído de [Farama, 2022].

Se ha optado por el uso del entorno FrozenLake por ser un entorno sencillo de visualizar, y por tener unos espacios de acciones y de estados de un tamaño suficiente, pero no excesivo. Este mismo algoritmo, por supuesto, puede utilizarse para entornos con espacios de estados y acciones mucho mayores.

4.3. Exploración vs explotación

Tras profundizar en el funcionamiento del algoritmo Q-Learning, es de gran importancia analizar los conceptos de exploración y explotación, así como su influencia en el algoritmo, por lo que es a este tema al que se dedica la sección actual.

En primer lugar, se conoce como explotación al acto de aprovechar el conocimiento obtenido para tomar la mejor decisión. Esencialmente, esto consiste, en el caso de Q-Learning, en seleccionar la acción con el máximo valor esperado, puesto que es la que el agente, desde su conocimiento del entorno, considera que es la mejor. Sin embargo, al iniciar el entrenamiento, el agente no tiene conocimiento alguno sobre el entorno, es

decir, no tiene conocimiento que explotar, y los valores de la tabla Q han sido generados arbitrariamente.

La exploración, por el contrario, consiste en explorar distintos estados y acciones para obtener mayor información sobre el entorno, que posteriormente pueda ser explotada. Por lo tanto, ambos elementos, exploración y explotación, son imprescindibles para el aprendizaje por refuerzo: sin exploración no se tiene conocimiento que explotar, pero sin explotación no se aprovecha el conocimiento adquirido. La exploración ayuda, además a evitar que el agente caiga en máximos locales, ya que sin ella el agente podría considerar que una determinada acción a_i es buena, y tomarla siempre, cuando puede existir una acción a_j que el agente aún no ha explorado, y que ofrece mejores resultados.

Un ejemplo para visualizar esto es pensar en una persona que se muda a una ciudad nueva, y desea encontrar su restaurante favorito. Podría elegir uno al azar y, si le gusta, continuar comiendo ahí, explotando su conocimiento. Sin embargo, es posible que el restaurante de al lado, que no llegó a probar, sea mucho mejor (al igual que podría ser mucho peor). El hombre debe, por tanto, probar restaurantes nuevos de vez en cuándo.

Sin embargo, en cada momento únicamente se puede explorar o explotar, pero no ambas. Es necesario, por tanto, encontrar un balance entre exploración y explotación que permita llevar a cabo el entrenamiento de la manera más eficiente posible. Esto es lo que se conoce como el dilema de la exploración-explotación, y es lo que da lugar a las distintas variantes de Q-Learning, que siguen la misma estructura general, pero utilizan estrategias diferentes para decidir si explorar o explotar. Se describen a continuación tres de las políticas de selección de acciones más importantes, y con las que se comparará la versión cuántica presentada en este trabajo, y descrita en la sección 4.4.

4.3.1. Épsilon-greedy

La política Épsilon-greedy es una de las más importantes por ser la primera política implementada, y se considera la estrategia "básica" de Q-Learning por su simplicidad. En ella se añade un nuevo hiperparámetro, ϵ, que regula la probabilidad de exploración: en cada momento, el agente tendrá una probabilidad ϵ de tomar una acción aleatoria (explorar) y una probabilidad $1 - \epsilon$ de tomar la que, en base al conocimiento, considera mejor (explotar).

Por supuesto, y al igual que ocurría con el resto de hiperparámetros del modelo, hallar un valor ideal de ϵ no es trivial, puesto que un valor demasiado alto no aprovecharía apenas

el conocimiento adquirido, mientras que un valor demasiado bajo sería muy susceptible de caer en extremos locales. Por esto, una práctica común es la de añadir un hiperparámetro más, generalmente denominado ϵ-decay (decaimiento de ϵ), que hace que ϵ varíe a lo largo del tiempo: al principio del entrenamiento, el conocimiento que tiene el agente es escaso y poco fiable, puesto que habrá visitado cada estado pocas veces, o incluso ninguna, por lo que es más efectivo realizar exploración que explotación; al final del entrenamiento, por el contrario, la mayoría de estados han sido visitados varias veces y el conocimiento del agente es sólido, por lo que se debe priorizar la explotación sobre la exploración. De este modo, el hiperparámetro ϵ-decay permite que ϵ vaya reduciéndose según el entrenamiento vaya avanzando, permitiendo una transición de un algoritmo principalmente explorador a uno principalmente explotador.

Existen muchos modos de implementar el decaimiento de ϵ, aunque generalmente se utiliza un decaimiento exponencial que permite que, independientemente de la longitud del entrenamiento, el valor nunca alcance el cero (ni valores inferiores), a diferencia de otros métodos como el decaimiento lineal.

La selección de acciones mediante ϵ-greedy tendría un aspecto similar al mostrado en el Algoritmo 2.

Algoritmo 2: Política de selección de acciones con ϵ-greedy

$r \leftarrow$ número aleatorio en $[0, 1]$;
if $r < \epsilon$ **then**
  $a \leftarrow$ acción seleccionada aleatoriamente
else
  $a \leftarrow \arg\max_a Q(e, a)$
end
return a

4.3.2. Softmax

De la definición de ϵ-greedy, puede inferirse qué uno de los motivos por el que es una de las políticas menos eficientes: cuando se realiza la exploración, todas las opciones son equiprobables. Esto significa que la mejor opción y la peor tienen exactamente la misma probabilidad de ser seleccionadas en la fase de exploración y, si bien todas las acciones deben ser exploradas, puede ser conveniente centrarse en las más prometedoras. De esta

forma, puede buscarse que, incluso en la fase de exploración, la acción que el agente considera la "segunda mejor" tenga mayor probabilidad de aparecer que la que considera la "tercera mejor", y así sucesivamente. Esto es lo que pretende resolver la política Softmax, que utiliza la función homónima que puede verse en la Ecuación (4.5), donde v representa el conjunto de valores de entrada y n el número de datos.

$$softmax(v_i) = \frac{e^{v_i}}{\sum_{k=1}^{n} e^{v_k}}. \tag{4.5}$$

Esta función toma como entrada una serie de valores y los normaliza entre 0 y 1 de manera que se preserva el orden, es decir, el dato que era mayor en los valores de entrada sigue siendo el mayor en la salida. El resultado de aplicar Softmax a los valores esperados de las distintas acciones se utiliza como distribución de probabilidad de cada elemento, es decir, la probabilidad de ejecutar la acción a_i en el estado e será $softmax(Q(e, a_i))$. De esta manera, si bien todas las acciones tienen una determinada probabilidad de ser seleccionadas, aquellas más prometedoras serán seleccionadas con mayor frecuencia.

Además, existe una variante de esta ecuación denominada Softmax con temperatura (4.6). La temperatura τ es otro hiperparámetro que se añade al modelo y que, de manera similar al caso de ϵ, regula la exploración/explotación afectando a la proporcionalidad de las probabilidades generadas: cuanto menor sea τ, más se acentuará la diferencia entre las probabilidades de salida, mientras que cuanto mayor sea τ, menor será esta diferencia. Así, un τ muy elevado puede llevar prácticamente a la exploración total, puesto que las acciones serán casi equiprobables independientemente del valor esperado; mientras que un τ pequeño lleva a una explotación casi total, puesto que las diferencias en los valores esperados se magnifican. Al igual que en con ϵ, se introduce generalmente otro hiperparámetro que regule la variación de τ con el tiempo, de manera que se transicione de una mayor exploración a una mayor explotación.

$$softmax(v_i) = \frac{e^{\frac{v_i}{\tau}}}{\sum_{k=1}^{n} e^{\frac{v_k}{\tau}}}. \tag{4.6}$$

4.3.3. Upper Confidence Bound

Puesto que los entornos en los que se utiliza Q-Learning son no-deterministas, los valores esperados no son exactos, y cuentan con una incertidumbre, que será menor cuantas más veces se visite el estado. Esto significa que el valor real de una determinada acción

puede estar en cualquier punto de este intervalo de confianza, que se calcula mediante métodos estadísticos (Figura 4.5). A diferencia de los métodos anteriores, que no trataban esta incertidumbre de manera directa, y simplemente consideraban que el valor no era preciso, en UCB el intervalo de confianza es una parte crucial del algoritmo. UCB, dentro de su optimismo, asume que el valor real de una acción está en el límite superior de su intervalo de confianza (de ahí su nombre, Upper Confidence Bound, o límite superior de confianza, en español), y por tanto selecciona la acción basándose en esto, y no en el valor esperado.

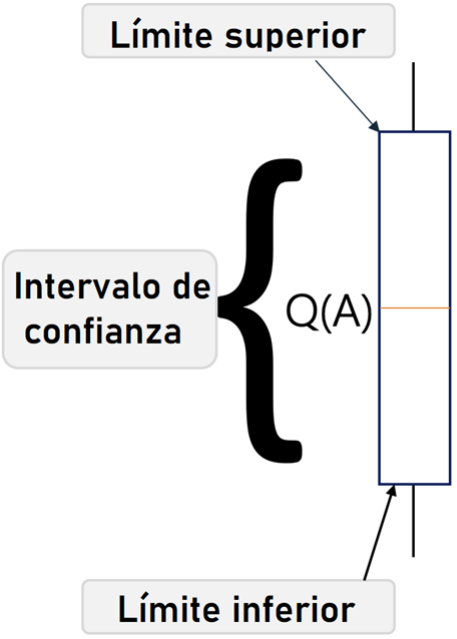


Figura 4.5: Representación del intervalo de confianza de UCB. Adaptado de [GeeksForGeeks, 2020].

Matemáticamente, esto se hace utilizando una función de bonus β que mide la distancia entre el valor esperado v_i de una acción a_i y el límite superior de su intervalo de confianza, que es el valor que se utilizará. El valor de β para un estado e y una acción a puede calcularse como

$$\beta(e, a) = C\sqrt{\frac{2\log t}{\sum_i N(e, a_i)}},$$

donde:

- t representa el número de pasos realizados hasta el momento.
- $N(e, a)$ representa el número de veces que se ha tomado la acción a desde el estado

e.

- C es un hiperparámetro, denominado *valor de confianza*, y que regula el nivel de exploración del algoritmo: para valores altos de C, se realizará una mayor proporción de exploración, mientras que si $C = 0$, se realiza una explotación completa.

Así, la acción seleccionada A se calcula como

$$A = \arg\max(Q(e, a) + \beta(e, a)),$$

esto es, considerando no el valor esperado de la acción, sino su límite superior de confianza.

Aunque, generalmente, C suele considerarse una constante, es posible incluir otro hiperparámetro que regule la variación de C, con el fin de que se transite de una mayor exploración a una mayor explotación (un valor de 1 en este hiperparámetro convertiría a C en una constante).

4.4. Descripción de la política de selección de acciones cuántica propuesta: QAASP

La naturaleza no-determinista de la cuántica plantea la posibilidad de aprovechar alguna de sus propiedades para el desarrollo de una política de selección de acciones. Si bien existen varios estudios que introducen versiones cuánticas del algoritmo Q-Learning utilizando puertas cuánticas, en este caso se propone el uso de Quantum Annealing como mecanismo para la selección de acciones. Para ello, se modela el proceso de selección de acciones como un problema QUBO.

4.4.1. Problemas QUBO

Los problemas de tipo QUBO (Quadratic Unconstrained Binary Optimization) son un conjunto de problemas que pueden ser resueltos mediante Quantum Annealing (entre otros métodos) y que, como su propio nombre indica, se caracterizan por ser:

- Cuadráticos, es decir, los términos de la función a minimizar tienen un grado no mayor que 2.
- Sin restricciones, por lo que por ejemplo no se pueden establecer restricciones del dominio de la función, ni incompatibilidades entre variables (por ejemplo, que cuando x_1 valga 1, x_2 valga 0).

- Binario, es decir, todas las variables del problema deben ser binarias, tomando los valores 0 o 1.
- Los problemas deben ser de Optimización, es decir, se debe buscar el mínimo de una función determinada.

Sin embargo, definir un problema de optimización sin restricciones resulta complicado, puesto que la mayoría de problemas reales cuentan con algún tipo de restricción. Por ello, existen estrategias para introducir restricciones dentro de los problemas QUBO, que esencialmente consisten en modificar la función mediante una penalización. Supongamos, por ejemplo, que dado un $x \in \{0,1\}^3$, se quiere minimizar una determinada función

$$f(x) = x_1x_2 + 5x_1x_3 + x_2$$

sujeta a una determinada restricción

$$r \equiv x_1 + x_2 = 1,$$

que representa una XOR entre x_1 y x_2, es decir, exactamente una de las dos debe valer 1. Sería posible reescribir esta restricción como $x_1 + x_2 - 1 = 0$ y, posteriormente, definir una nueva función

$$g(x) = f(x) + \lambda r = x_1x_2 + 5x_1x_3 + x_2 + \lambda(x_1 + x_2 - 1).$$

Vemos como en esta función se introduce un parámetro de penalización λ que multiplica a la restricción. De esta manera, siempre que la restricción se cumpla, el término λr valdrá 0. Si no se cumple, sin embargo, se sumará a la función un factor λ que, al aumentar el valor de la misma, hará que esa combinación deje de ser mínimo, y por tanto no se seleccione. Este efecto se ve reforzado cuanto más grande sea el valor de λ, puesto que cuanto mayor sea λ más ineficiente resultará incumplir las restricciones. Sin embargo, se debe tener en cuenta que, a pesar de que en la función no hay restricciones sobre los coeficientes a utilizar, a la hora de ejecutar el algoritmo en un computador cuántico, estos irán normalizados, generalmente entre 0 y 1. Esto significa que, si la diferencia entre el valor de λ y los coeficientes de la función es demasiado grande, al normalizar, estos últimos serán prácticamente insignificantes, puesto que la mayor parte del "peso" de la función final la tiene λ, haciendo que todos los estados que cumplan las restricciones tengan niveles de energía muy similares, y que por tanto la optimización sea menos eficiente, puesto que

elegir una "mala" combinación (que cumpla las restricciones) es casi igual de bueno que elegir la combinación óptima. Por ello, en general debe buscarse un valor de λ que sea lo más pequeño posible mientras "obligue" al sistema a cumplir las restricciones. Por ejemplo, podría calcularse cuál es el máximo beneficio que se podría obtener incumpliendo las restricciones y utilizar un λ ligeramente superior al valor calculado. Es importante, sin embargo, asegurarse de que la restricción que multiplica a λ sea siempre positiva puesto que, de lo contrario, existirían casos en los que incumplir la restricción reduciría el valor de la función en lugar de aumentarlo. Por ello, si el problema lo requiere, es común elevar al cuadrado la restricción, para garantizar que es positivo. De este modo, la función final a minimizar h sería:

$$g(x) = f(x) + \lambda r^2 = x_1x_2 + 5x_1x_3 + x_2 + \lambda(x_1 + x_2 - 1)^2.$$

Esto no es necesario si, por la definición de la restricción, ninguna combinación de valores puede hacer que el multiplicador de λ sea negativo.

Por supuesto, es posible añadir varias restricciones utilizando este mismo sistema, y es posible utilizar λs diferentes para cada una de ellas.

4.4.2. Política de Selección de Acciones como un problema QUBO

Una vez comprendido el funcionamiento de los problemas QUBO, es posible implementar una política de selección de acciones utilizando una definición de este tipo. Para ello, supónganse una serie de acciones a_i cuyos valores esperados son v_i. Se codifica el problema creando un conjunto de variables x tal que:

$$x_i = \begin{cases} 1 & \text{si se selecciona la acción } a_i, \\ 0 & \text{en otro caso.} \end{cases}$$

Es decir, x indica la acción a tomar. Entonces, puede definirse una función

$$f(x) = -\sum_i v_i x_i$$

en la que se multiplica cada acción x_i por su valor esperado v_i, de modo que aquella acción con mayor valor esperado devolverá el máximo valor. Además, puesto que el Quantum Annealing únicamente es capaz de minimizar, y el objetivo del problema es maximizar el valor esperado, se invierte su signo para convertirlo en un problema de minimización.

Esta función, sin embargo, debe estar sujeta a una restricción, y es que únicamente puede seleccionarse una acción. Por tanto, siguiendo el procedimiento detallado anteriormente, se formula la restricción como

$$\sum_i x_i = 1,$$

que indica que exactamente un valor de x_i puede estar "activo" y, a continuación, se reescribe como

$$\sum_i x_i - 1 = 0.$$

Puesto que, si no se seleccionara ninguna variable, el valor de la restricción sería negativo, será necesario elevarlo al cuadrado al generar la función resultante:

$$g(x) = -\sum_i v_i x_i + \lambda \left(\sum_i x_i - 1 \right)^2 .$$

A priori, podría parecer que esta función únicamente se dedica a seleccionar la acción con el máximo valor esperado, es decir, que explota únicamente, sin explorar. Sin embargo, debe tenerse en cuenta la naturaleza aleatoria de la cuántica, que hace que cualquiera de las acciones sea posible, si bien serán más probables aquellas con un mayor valor esperado.

Además, se ha introducido una restricción en la misma y, como consecuencia, un parámetro λ. Este parámetro, que pasa a convertirse en un hiperparámetro del modelo, puede utilizarse para regular la diferencia de probabilidades de las distintas acciones puesto que, como se ha detallado en la sección 4.4.1, cuando el valor de λ es muy elevado, los coeficientes de la función pierden importancia, de modo que, en el caso extremo en el que $\lambda \to \infty$, todas las acciones son equiprobables. Esto puede verse en la Figura 4.6[5], en la que se muestra cómo varía la distribución de probabilidades al variar el multiplicador de λ. En cierto modo, el efecto del hiperparámetro λ es similar al de la temperatura en Softmax. A diferencia de la temperatura, sin embargo, se debe tener especial cuidado con el valor de λ seleccionado, puesto que, si este es demasiado bajo, no se cumplirá la restricción establecida, y se seleccionará más de una acción (lo cual carece de sentido).

[5]Al observarse esta Figura, no debe cometerse el error de pensar que el primer valor esperado (-1) corresponde a la primera "barra" del gráfico, el segundo con la segunda, etc. sino que se corresponden, respectivamente, con los estados 0001, 0010, 0100 y 1000.

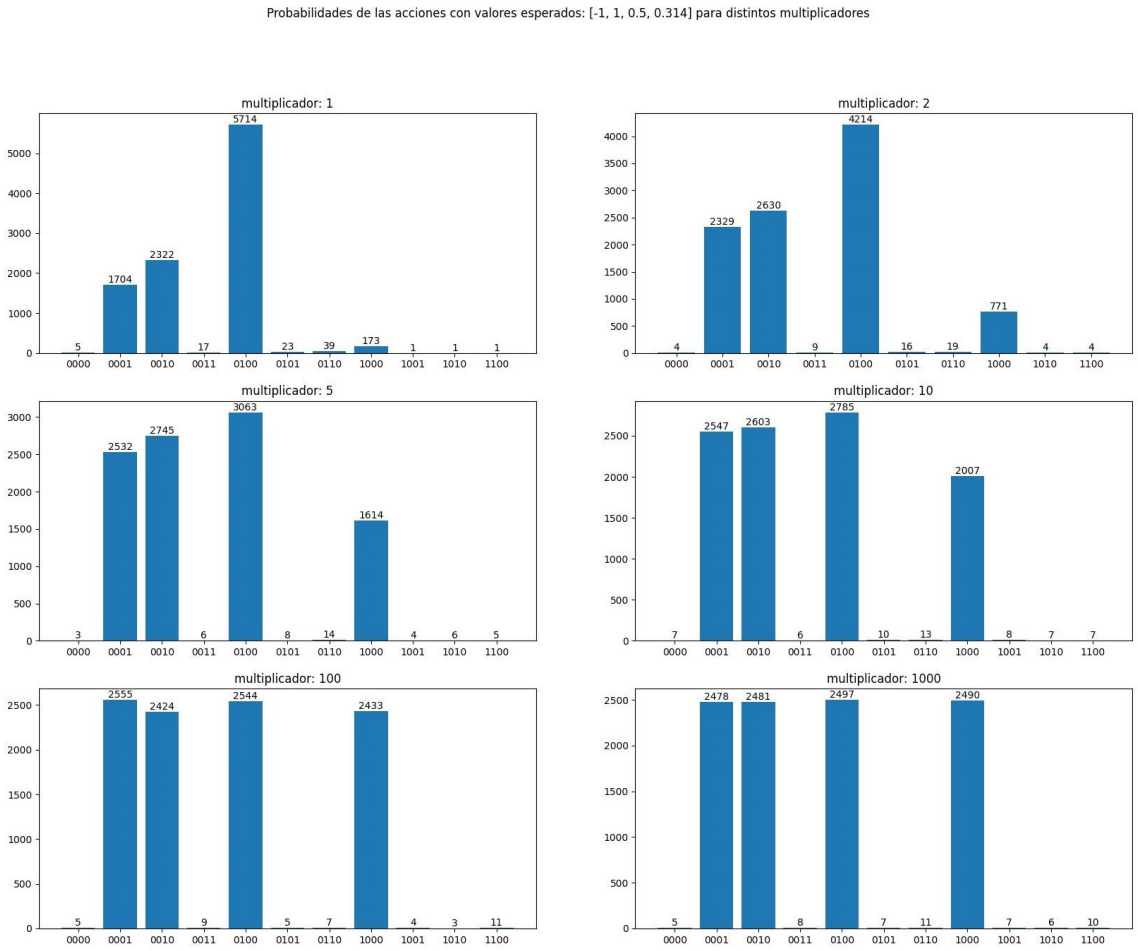


Figura 4.6: Representación de las probabilidades de cada una de las acciones según el multiplicador m utilizado. Los valores esperados para las acciones son, respectivamente, -1, 1, 0.5 y 0.314.

Es interesante, al igual que en casos anteriores, añadir un segundo hiperparámetro que regule el decaimiento de λ, de manera que se pueda iniciar el entrenamiento con un valor alto del λ que propicia una mayor exploración, y con el tiempo reducirlo para obtener una mayor proporción de explotación, siempre teniendo en cuenta los valores mínimos bajo los que no debe caer λ.

Se debe tener en cuenta, además, que el efecto del valor de λ depende de la relación entre los valores de los coeficientes y el del propio λ. Esto significa que un $\lambda = 1$ puede suponer, en una determinada fase del entrenamiento, mucha exploración si los valores esperados se encuentran en un rango como $[0, 0.05]$, mientras que si se encuentran en el rango $[0, 1]$ el mismo valor de λ supondrá principalmente explotación. Por ello, en lugar de utilizar un valor de λ arbitrario, se utiliza un multiplicador m sobre un valor de base λ_b que

se calcula, simplemente, como la suma de los valores absolutos de los valores esperados, multiplicados por un coeficiente $c > 1$, es decir,

$$\lambda = m\lambda_b,$$

donde

$$\lambda_b = c\left(\sum_i |v_i|\right),$$

siendo v_i los valores esperados. De esta manera, λ se mantiene proporcional a los valores esperados en todo momento, de modo que se mantiene el nivel de exploración-explotación deseado independientemente del rango de valores esperados. c representa el "margen" que tiene λ respecto del máximo valor posible de la función, es decir, si $c = 1.3$, incumplir la restricción se castiga con un valor un 30 % superior al valor más óptimo alcanzable por la función. Por ello, m no debería ser nunca inferior a 1, puesto que de serlo, no se podría garantizar que se cumpla la restricción.

Con esto, se obtiene una política de selección de acciones que utiliza las propiedades cuánticas de la materia y el Quantum Annealing para intentar resolver el dilema de la exploración-explotación. Esta política será denominada, por simplicidad, QAASP: Quantum Annealed Action Selection Policy. A continuación, se discutirá la implementación que se ha hecho de esta política, con el fin de realizar pruebas de rendimiento comparativas con las variantes clásicas del algoritmo descritas en la sección 4.3.

4.5. Implementación de la política de selección de acciones

Para la implementación de QAASP se ha utilizado, como base, la aplicación Q-Learning desarrollada en [Caponera De Cobellis, 2021], que se ofrece como código abierto en GitHub. Esta aplicación ofrece una implementación genérica del algoritmo Q-Learning, y está específicamente diseñada para permitir la introducción de nuevas políticas de modo sencillo. Sin embargo, con el fin de diseñar un código que pueda ser ejecutado sobre un ordenador cuántico real sin necesidad de modificaciones, se ha extraído el código genérico de Q-Learning de la aplicación, e importado a un nuevo proyecto sin interfaz gráfica que es compatible con D-Wave Leap, una de las plataformas más utilizadas para la ejecución de problemas de Quantum Annealing sobre ordenadores cuánticos reales. Gracias al uso del código de esta aplicación, tan solo es necesario crear una nueva clase `QAASP` que ex-

tienda a la clase abstracta `Politica`, y redefinir algunos métodos, principalmente el de `seleccionar_accion`, cuya implementación se muestra en el Código 1.

Para la implementación, se ha utilizado la librería `qubovert`, que simplifica la implementación de los problemas QUBO. Así, en la línea 8 se define un diccionario `x` en el que se relaciona cada variable `x[i]` con su índice `i` y, posteriormente, en las líneas 13-15 se define la función base a optimizar. A continuación, en las líneas 18 y 19 se genera la restricción, que exige que la suma de todas las variables `x[i]` valga exactamente 1, y se le asigna un valor de `lam` a λ, que ya ha sido calculado previamente en las líneas 2-5. Para introducir la restricción se utiliza la función de qubovert `add_constraint_eq_zero`, que permite especificar una restricción en el mismo formato que se definió en secciones anteriores de este trabajo, es decir, como una función igualada a 0. Posteriormente, las líneas 22 y 23 convierten el problema definido, que hasta ahora se encuentra formulado en un formato interno de qubovert, a un problema QUBO apto para ser resuelto en los ordenadores de D-Wave. La línea 28 utiliza un simulador para ejecutar los cálculos. Sin embargo, ejecutar el algoritmo en un ordenador cuántico real tan solo requiere modificar dicha línea, sustituyendo el `SimulatedAnnealingSampler` por la clase correspondiente para la ejecución en máquinas reales. Finalmente, se extraen los resultados y se descodifica la acción seleccionada, que será devuelta por la función.

4.6. Análisis del rendimiento

En el caso de los algoritmos híbridos clásicos-cuánticos, incluyendo a QAASP, es importante distinguir entre el rendimiento en términos de tiempo de ejecución, y en términos de iteraciones/pasos, y es que, generalmente, los algoritmos híbridos conllevarán un tiempo de ejecución considerablemente más largo que los algoritmos puramente clásicos (o puramente cuánticos), por motivos relacionados con el *overhead*, es decir, el coste de transmitir la información desde el procesador clásico al cuántico, y viceversa. Esto, como se podrá deducir, resulta mucho más costoso que realizar una o varias operaciones en el mismo procesador, sobre todo si no son operaciones muy complejas. En algunos casos de Quantum Annealing, se ha observado que el overhead del procesador clásico al cuántico puede ser 72 veces más costoso que el propio annealing [Fernández-Villaverde and Hull, 2023], reafirmando que la conexión entre estos dos sistemas requiere aún muchas mejoras. Por ello, en este trabajo se hará una comparativa centrada en el número de iteraciones o pasos

```python
    def seleccionar_accion(self):
        c = 1.1 # se considera constante del modelo
        estado = self.agente.estado
        lambda_b = self.parametro
        lam = np.sum(self.agente.Q[estado])*lambda_b*c

        #crear las variables
        x = {i: boolean_var(f'x_{i}') for i in range(self.n_acciones)}

        #construir el modelo
        model = 0
        # función a minimizar
        for i in range(self.n_acciones):
            valor_esperado = self.agente.Q[estado][i]
            model -= x[i] * valor_esperado

        # Solo una acción
        restriccion = [x[i] for i in range(self.n_acciones)]
        model.add_constraint_eq_zero(sum(restriccion) - 1, lam=lam)

        #Convertir a QUBO
        qubo = model.to_qubo()
        dwave_qubo = qubo.Q

        #Ejecutar
        n_seleccionadas = 0
        while n_seleccionadas!=1:
            res = SimulatedAnnealingSampler().sample_qubo(dwave_qubo, num_reads=1)
            qubo_solution = res.first.sample
            model_solution = model.convert_solution(qubo_solution)

            #Filtrar las acciones que han sido seleccionadas
            seleccionadas = list(filter(lambda s: s[1] != 0, model_solution.items()))
            n_seleccionadas = len(seleccionadas)

        #Obtener y devolver la acción seleccionada
        accion = int(seleccionadas[0][0].split('_')[1])

        return accion
```

Código 1: Fragmento de código en Python que implementa la selección de acciones mediante Quantum Annealing. Para ello, hace uso de distintas librerías como *numpy*, *qubovert* o *dwave-neal*.

del entrenamiento, independientemente del tiempo requerido para cada paso.

Para la medida del rendimiento, se ha utilizado el entorno FrozenLake 4x4 ejecutando el entrenamiento 20 veces por cada algoritmo. Nótese que, en este entorno, y según la documentación oficial, se considera que el entrenamiento finaliza cuando en los últimos 100 episodios se obtiene una recompensa media igual o superior a 0.78 (esto es, alcanzar el objetivo el 78 % de las veces). Sin embargo, para evitar tiempos de ejecución excesivamente largos, es necesario fijar un límite superior de episodios tras los cuales detener el entrenamiento, que en este caso se ha situado en 2000. Pueden verse los resultados de estas prueba en la Figura 4.7.

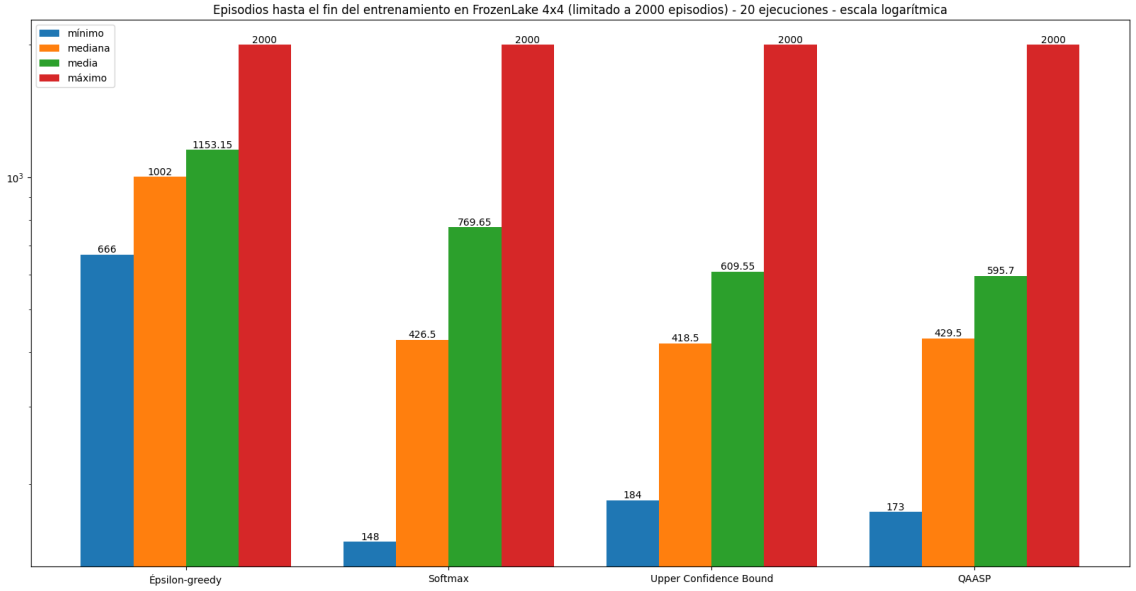


Figura 4.7: Diagrama de barras que muestra el rendimiento del algoritmo en el entorno FrozenLake 4x4. 20 ejecuciones. Escala logarítmica.

En este gráfico se muestran, para cada algoritmo, la ejecución más corta (mínimo número de episodios), la más larga (máximo número de episodios), la ejecución promedio, y la mediana de los episodios de las ejecuciones. Lo primero que se observa es que todos los algoritmos han alcanzado los 2000 episodios fijados como límite superior, es decir, en todos los algoritmos, en al menos una ejecución se habrían requerido más de 2000 episodios. Además se ve cómo, en las tres métricas restantes (mínimo, mediana y media) el algoritmo ϵ-greedy tiene un rendimiento notablemente inferior al resto de algoritmos (notar que el gráfico está en escala logarítmica, por lo que, si bien visualmente pueda parecer que la diferencia no es tan grande, realmente sí lo es), lo cual es lógico si se considera que es

la política más simple de todas. Los restantes algoritmos, por otro lado, se encuentran bastante cercanos en el resto de valores. Los valores más importantes son el de la media y la mediana, puesto que el valor mínimo es el más influenciable por el azar. Se puede observar que, en la media, tanto UCB como SoftMax se encuentran unos 170 episodios por debajo, aproximadamente, lo cual supone una diferencia significativa. La diferencia entre estos dos últimos, sin embargo, es de apenas unos 15 episodios. En la mediana, por el contrario, se puede ver cómo la diferencia es mucho menor entre los tres, siendo la diferencia entre el menor (Softmax) y el mayor (QAASP) de apenas 11 episodios.

Debe tenerse en cuenta, sin embargo, que el valor del promedio se ve fuertemente influenciado por el límite superior fijado de 2000 episodios, puesto que afecta del mismo modo una ejecución que hubiera tardado 2001 episodios que una que hubiera tardado 100.000 (ambas se consideran 2000). Por tanto, es razonable pensar que en otras circunstancias el valor promedio será superior. Si bien es inabarcable ejecutar el algoritmo sin un límite superior, puesto que podría tardar una cantidad de tiempo arbitrariamente grande, sí es posible ampliar este límite hasta los 10.000 episodios, lo que proporciona una visión más fiable de los datos. Puede verse esto mismo en la Figura 4.8, en el que se entrena cada algoritmo 100 veces (en lugar de 20) con el fin de minimizar lo más posible el efecto del azar.

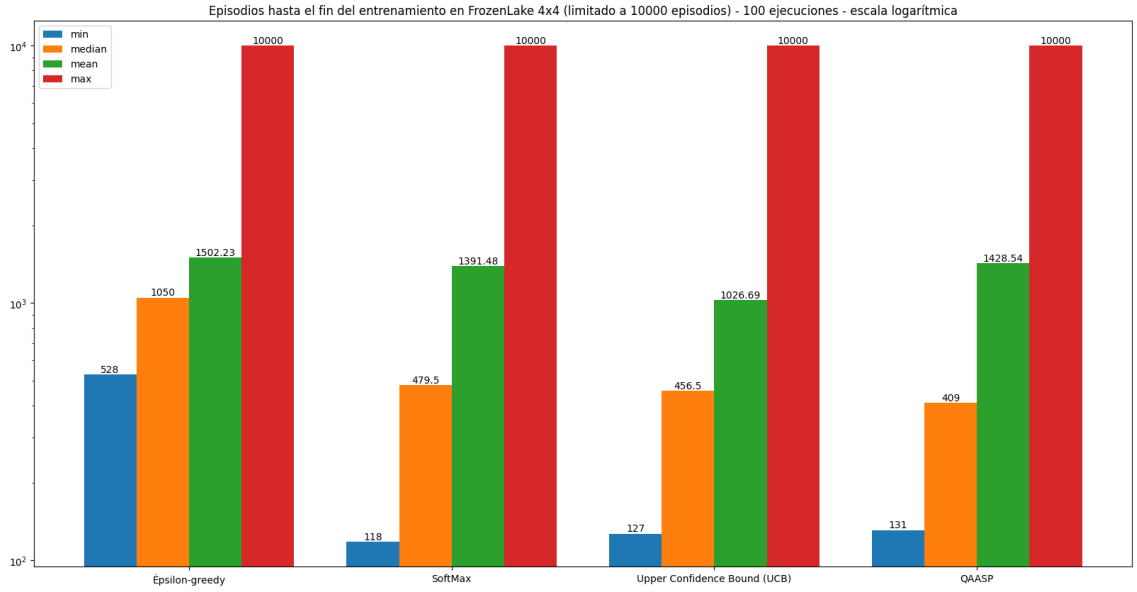


Figura 4.8: Diagrama de barras que muestra el rendimiento del algoritmo en el entorno FrozenLake 4x4. 100 ejecuciones. Escala logarítmica.

En esta figura se observa, tal y como se planteó anteriormente, que el promedio aumenta significativamente en todos los casos. Del mismo modo, al haberse realizado más ejecuciones, disminuye el valor de la ejecución mínima, por motivos puramente estadísticos. Sin embargo, al observar la mediana, se puede ver cómo esta aumenta en todos los casos excepto en QAASP, que disminuye, a pesar de tener un promedio considerablemente superior a UCB y Softmax. Esto significa que, si bien en la ejecución promedio QAASP será más lento que los otros dos algoritmos, el 50 % de las veces terminará más rápido o, visto de otro modo, se puede interpretar como que, en ocasiones, el algoritmo toma decisiones equivocadas que lo llevan a "perderse" y a ejecutar un entrenamiento excesivamente largo, lo cual lleva a una mayor dispersión de los datos, especialmente en la parte superior a la mediana. Con el fin de observar mejor esto, en las Figuras 4.9 y 4.10 se muestran unos gráficos box-and-whiskers[6] con los mismos datos, utilizando escala logarítmica y outliers en el primero, y escala lineal sin outliers en el segundo.

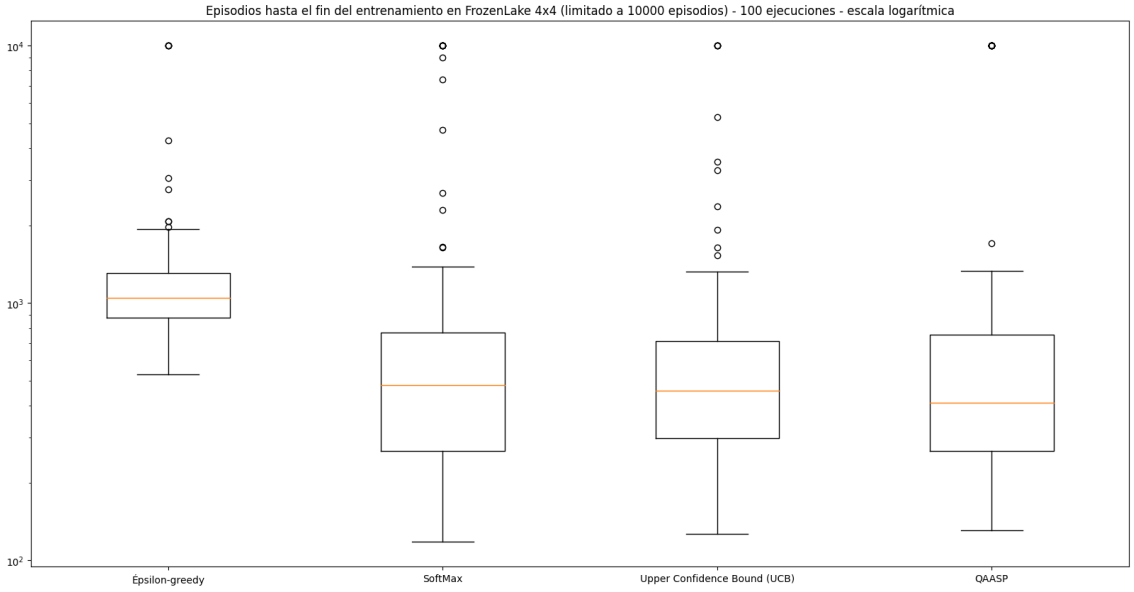


Figura 4.9: Gráfico box-and-whiskers que muestra el rendimiento del algoritmo en el entorno FrozenLake 4x4. 100 ejecuciones. Escala logarítmica.

[6]Este tipo de gráfico muestra la dispersión de los datos: los espacios entre los "bigotes" y la caja representan, respectivamente, el primer cuartil (25 % más bajo de datos) y el último (25 % más alto), mientras que la caja representa el rango intercuartílico, es decir, el 50 % de datos "centrales". La línea naranja representa la mediana, es decir, separa el 50 % más bajo de datos del 50 % más alto. Los círculos por encima o por debajo de los bigotes representan los datos atípicos, es decir, los que son anormalmente altos o bajos. Se consideran outliers los datos que se encuentren 1.5 veces el rango intercuartílico por encima del tercer cuartil o por debajo del primero.

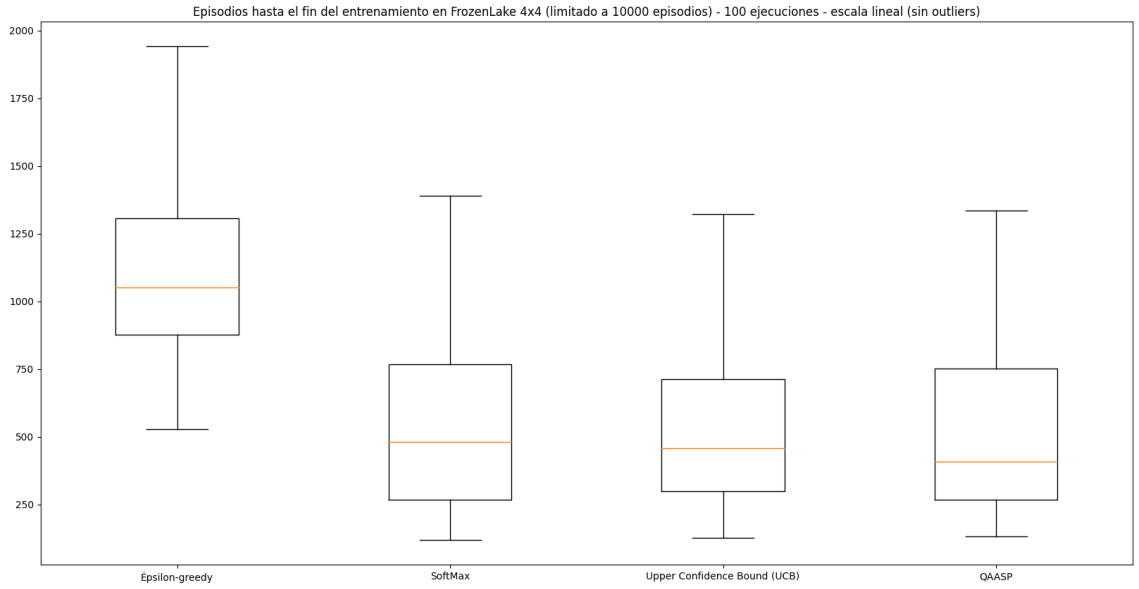


Figura 4.10: Gráfico box-and-whiskers que muestra el rendimiento del algoritmo en el entorno FrozenLake 4x4. 100 ejecuciones. Escala lineal, sin outliers.

Se puede ver en la Figura 4.9 que, en comparación al resto, QAASP cuenta con considerablemente menos outliers, y esto se debe principalmente a que se "pierde" con relativa frecuencia, obteniendo valores altos (al suceder a menudo dejan de ser datos atípicos - outliers - para ser datos normales). Además, en la Figura 4.10 se aprecia cómo, efectivamente, el rango intercuartílico (altura de la "caja") es más amplio que el de UCB, si bien su mediana es más baja, se modo que se ve cómo el tercer cuartil es considerablemente mayor que el segundo o el primero. Retomando la interpretación en la que se plantea que el algoritmo se "pierde", podría decirse que, en comparación con los otros algoritmos, se "pierde" con mayor frecuencia, pero cuando no se pierde es más efectivo (mediana menor). Un modo de aprovechar esto sería fijar un límite superior ligeramente mayor que la mediana y que, al ser superado, se considere que el algoritmo se ha "despistado", por lo que se reinicia el entrenamiento[7]. De este modo, se consigue "reconducir" al algoritmo cuando se "pierde" y, estadísticamente, una de cada dos ejecuciones debería encontrarse por debajo de los 409 episodios por lo que, incluso en el caso de reiniciar el entrenamiento, este debería finalizar en menos de 818 episodios, considerablemente inferior a la ejecución promedio. Incluso si fuera necesario ejecutar el entrenamiento tres veces, el cómputo total de episodios sería menor que el promedio. Si bien este procedimiento sería igualmente

[7]Por supuesto, esto es tan solo una descripción muy simple, pero podrían tenerse en cuenta otros factores como qué tan lejos está del fin del entrenamiento para decidir si reiniciarlo o no.

válido para otros algoritmos como UCB, QAASP se vería especialmente beneficiado por tener una diferencia tan grande entre el promedio y la mediana.

Sin embargo, al ver cómo los descriptores estadísticos varían significativamente al variar el tamaño de la muestra, cabe plantear un análisis estadístico que tenga en cuenta los intervalos de confianza para comparar las medias. Para ello, se calcula el intervalo de confianza de la diferencia de medias entre QAASP y el resto de algoritmos, haciendo uso de la fórmula establecida para ello [Pintarelli, 2023][Canavos, 2003]:

$$lb = \overline{X_1} - \overline{X_2} - t_{\alpha/2, n_1+n_2-2} S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$$

y

$$ub = \overline{X_1} - \overline{X_2} + t_{\alpha/2, n_1+n_2-2} S_p \sqrt{\frac{1}{n_1} + \frac{1}{n_2}},$$

siendo:

- lb el límite inferior (lower bound) y ub el límite superior (upper bound) del intervalo de confianza
- $\overline{X_1}$ y $\overline{X_2}$ las medias de las muestras a comparar
- n_1 y n_2 los tamaños de las muestras
- $1 - \alpha$ el nivel de confianza
- $t_{a,b}$ el valor de la distribución t-student para el nivel de confianza a y b grados de libertad
- S_p un estimador combinado de las varianzas comunes de ambas muestras, calculado como

  $$S_p = \sqrt{\frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}},$$

  donde S_1 y S_2 representan la varianza muestral de cada muestra.

Utilizando, por tanto, la fórmula anterior, tomando como X_1 los datos de QAASP, y como X_2 los de los otros algoritmos, y $\alpha = 0.05$ (nivel de confianza 95 %), se obtienen los siguientes resultados:

- El intervalo de confianza para $\mu_{\text{QAASP}} - \mu_{\epsilon\text{-greedy}}$ es: (-712, 654).
- El intervalo de confianza para $\mu_{\text{QAASP}} - \mu_{\text{softmax}}$ es: (-705, 779).

- El intervalo de confianza para $\mu_{\text{QAASP}} - \mu_{\text{UCB}}$ es: (-251, 1055).

Como se puede ver, los tres intervalos contienen al cero. Lo que, desde un punto de vista estadístico, implica que no puede afirmarse con rotundidad que el promedio de QAASP sea distinto al de cualquiera de los otros algoritmos. Esto no implica necesariamente que los tres algoritmos tengan el mismo promedio y, de hecho, observando los gráficos anteriores, sería esperable que no fuera así; sin embargo, no se tiene la suficiente evidencia para asegurarlo. Es interesante destacar la amplitud que tienen los intervalos de confianza calculados, que están entorno a los 1400 episodios.

Para fundamentar esto, se ha utilizado una prueba de hipótesis para cada caso, del que se derivan los p-valores (p-values, en inglés) asociados, que deben ser estudiados con el fin de evaluar la solidez de la afirmación anterior. El p-valor mide la probabilidad de que la diferencia de medias en las distribuciones se deba al azar, y no a una diferencia en la media real de la población. Tras efectuar el cálculo de los valores, se obtiene que:

- El p-valor para $\mu_{\text{QAASP}} - \mu_{\epsilon\text{-greedy}}$ es 0.81.
- El p-valor para $\mu_{\text{QAASP}} - \mu_{\text{softmax}}$ es 0.91.
- El p-valor para $\mu_{\text{QAASP}} - \mu_{\text{UCB}}$ es 0.22.

Esta métrica permite medir la robustez de la decisión de rechazar la hipótesis nula (es decir, que las medias sean diferentes), de modo que un p-valor mayor implica una mayor probabilidad de haberse equivocado si se hubiera aceptado dicha hipótesis. Por ejemplo, se puede ver cómo el p-valor para ϵ-greedy es 0.81, lo que significa que la probabilidad de error al afirmar que las medias son diferentes es del 81 %. De modo análogo, las probabilidades de equivocarnos al afirmar que hay una diferencia de medias son del 91 % y 22 % respectivamente para softmax y UCB. Esto permite afirmar con mayor contundencia lo visto anteriormente mediante intervalos de confianza.

A la vista de los datos presentes, resulta difícil afirmar con rotundidad qué algoritmo es mejor, puesto que cada uno ofrece ventajas diferentes. No debe olvidarse, sin embargo, a la hora de implementar cualquiera de ellos, que, si el tiempo de ejecución es importante para el proyecto, QAASP y otros algoritmos híbridos pueden resultar considerablemente más lentos por la necesidad de interacción entre el computador clásico y el cuántico. En general, la eficiencia de los algoritmos dependerá de cada entorno y de sus características, por lo que, para cada proyecto, será necesario un análisis previo que evalúe qué algoritmo resulta más conveniente.

5. Conclusiones y Trabajo Futuro

En este trabajo se ha explorado una posible aplicación del Quantum Annealing al ámbito del machine learning y el aprendizaje por refuerzo. Como se ha podido ver, los resultados en el entorno FrozenLake son similares a los que ofrecen otros algoritmos clásicos como SoftMax, y esto se debe, en parte, a que la idea de base es muy similar. Al igual que ocurre en muchos otros ámbitos del Quantum Machine Learning, este algoritmo no pretende sustituir versiones clásicas del mismo, sino ser una opción más que considerar, una variante del algoritmo que, según el entorno, podrá ofrecer un rendimiento mejor o peor que las contrapartidas clásicas. Y es que, a diferencia de lo que se podría pensar, la computación cuántica y el Quantum Machine Learning no tienen como objetivo sustituir a la computación clásica, sino complementarla y ofrecer herramientas nuevas con posibilidades aún inexploradas. La política introducida en este trabajo, QAASP, es tan solo una primera aproximación del Quantum Annealing al aprendizaje por refuerzo, pero aún pueden desarrollarse muchos avances en este sentido. Algunas líneas de trabajo futuro incluyen, por ejemplo, el desarrollo de algoritmos cuánticos que no hagan uso de λ como hiperparámetro para regular la exploración/explotación, sino que utilicen otros métodos que pudieran resultar interesantes en otros ámbitos. Además, puede plantearse la idea de una versión completamente cuántica del algoritmo Q-Learning, codificando el propio entorno y las funciones de transición y recompensa dentro del mismo problema. Esto reduciría enormemente los tiempos de ejecución, puesto que se elimina el overhead entre los procesadores clásico y cuántico. Sin embargo, el diseño de un circuito que codifique el entorno requiere de un conocimiento sobre el mismo que el desarrollador debe poseer de antemano, por lo que esta versión del algoritmo podría no adaptarse adecuadamente a todas las situaciones.

Por otro lado, esta misma versión del algoritmo requiere un estudio en mayor profundidad, probando distintos tipos de entornos con distintos tamaños de los espacios de estados y de acciones. De este modo, será posible determinar con mayor precisión la calidad del mismo, y averiguar cómo se comporta en relación al tamaño del problema, para determinar en qué casos puede ser recomendable utilizarlo, y en qué casos resulta conveniente aplicar una versión clásica de Q-Learning.

Estas son tan solo algunas de las posibles líneas de trabajo futuras que pueden surgir de este proyecto, aunque pueden existir muchas otras, especialmente teniendo en cuenta

la gran cantidad de trabajo que aún falta por hacer en el campo del Quantum Machine Learning en general, y del Quantum Reinforcement Learning en particular.

Bibliografía

[Ayodele, 2010] Ayodele, T. O. (2010). X types of machine learning algorithms.

[Barty et al., 2007] Barty, K., Girardeau, P., Roy, J.-S., and Strugarek, C. (2007). Q-learning with continuous state spaces and finite decision set. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 346–351.

[Bonner, 2007] Bonner, A. (2007). *The art and logic of Ramon Llull a user's guide*. Brill.

[Caltech, 2023] Caltech (2023). What is superposition and why is it important? `https://scienceexchange.caltech.edu/topics/quantum-science-explained/quantum-superposition`.

[Canavos, 2003] Canavos, G. C. (2003). *Probabilidad y Estadística: Aplicaciones y métodos*. McGraw-Hill.

[Caponera De Cobellis, 2021] Caponera De Cobellis, R. R. (2021). Comparación de algoritmos de aprendizaje por refuerzo basados en q-learning.

[Carrizo Zirit, 2019] Carrizo Zirit, F. (2019). ¿es un hot dog un sándwich? `https://qr.ae/pyXw44`.

[Cervera-Lierta, 2018] Cervera-Lierta, A. (2018). Quantum annealing. `https://medium.com/@quantum_wa/quantum-annealing-cdb129e96601`.

[clwainwright, 2019] clwainwright, h. (2019). Understanding the role of the discount factor in reinforcement learning. Cross Validated. URL:`https://stats.stackexchange.com/q/428157` (version: 2019-09-21).

[Cole, 2020] Cole, D. (2020). The chinese room argument. `https://plato.stanford.edu/entries/chinese-room/#3`.

[Das, 2017] Das, A. (2017). The very basics of reinforcement learning. `https://becominghuman.ai/the-very-basics-of-reinforcement-learning-154f28a79071`.

[Farama, 2022] Farama (2022). Frozenlake (gymnasium). `https://gymnasium.farama.org/environments/toy_text/frozen_lake/`.

[Fernández-Villaverde and Hull, 2023] Fernández-Villaverde, J. and Hull, I. (2023). Dynamic programming on a quantum annealer: Solving the rbc model. *SSRN Electronic Journal*.

[GeeksForGeeks, 2020] GeeksForGeeks (2020). Upper confidence bound in reinforcement learning. https://www.geeksforgeeks.org/upper-confidence-bound-algorithm-in-reinforcement-learning/.

[Google, 2021] Google (2021). Lamda: our breakthrough conversation technology. https://blog.google/technology/ai/lamda/.

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

[Heidenreich, 2018] Heidenreich, H. (2018). What are the types of machine learning? https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f.

[Hunter and Rhodius, 2015] Hunter, R. L. and Rhodius, A. (2015). *Argonautica: Book IV*. Cambridge University Press.

[Martins, 2022] Martins, A. (2022). El "cerebro cuántico", la audaz teoría que puede ayudar a resolver el misterio de cómo surge la conciencia humana. https://www.bbc.com/mundo/noticias-64065872.

[Millán Páramo et al., 2014] Millán Páramo, C., Begambre Carrillo, O., and Millán Romero, E. (2014). Propuesta y validación de un algoritmo simulated annealing modificado para la solución de problemas de optimización. https://www.elsevier.es/es-revista-revista-internacional-metodos-numericos-calculo-338-articulo-propuesta-va

[Muller, 2022] Muller, A. (2022). What is quantum entanglement? a physicist explains einstein's "spooky action at a distance". https://www.astronomy.com/science/what-is-quantum-entanglement-a-physicist-explains-einsteins-spooky-action-at-a-dista

[Na8, 2017] Na8 (2017). Qué es overfitting y underfitting y cómo solucionarlo. https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/.

[OpenAI, 2021] OpenAI (2021). Frozenlake. https://gym.openai.com/envs/FrozenLake8x8-v0/.

[OpenAI, 2022] OpenAI (2022). Introducing chatgpt. https://openai.com/blog/chatgpt.

[Paolucci, 2020] Paolucci, R. (2020). A casual guide to dirac notation. https://towardsdatascience.com/a-casual-guide-to-dirac-notation-17961670ae7a.

[Pickel, 2021] Pickel, D. (2021). Supervised vs unsupervised learning - what's the difference? https://www.g2.com/articles/supervised-vs-unsupervised-learning.

[Pintarelli, 2023] Pintarelli, M. B. (2023). Intervalos de confianza. https://www.mate.unlp.edu.ar/practicas/117_12_12092016211213.pdf.

[Rocca, 2019] Rocca, J. (2019). Introduction to markov chains. https://towardsdatascience.com/brief-introduction-to-markov-chains-2c8cab9c98ab.

[Santana Vega, 2019] Santana Vega, C. (2019). Así funciona deepnude, la ia que te desnuda - (cgans y pix2pix). https://youtu.be/ysEjAqnHp64?t=1526.

[Shyalika, 2019] Shyalika, C. (2019). A beginners guide to q-learning. https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c.

[Silver et al., 2017] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., and et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359.

[TheGuardian, 2022] TheGuardian (2022). Google fires software engineer who claims ai chatbot is sentient. https://www.theguardian.com/technology/2022/jul/23/google-fires-software-engineer-who-claims-ai-chatbot-is-sentient.

[Trixler, 2013] Trixler, F. (2013). Quantum tunnelling to the origin and evolution of life. *Current organic chemistry*, 17:1758–1770.

[TURING, 1950] TURING, A. M. (1950). I.—computing machinery and intelligence. *Mind*, LIX(236):433–460.

[Wallace, 1995] Wallace, R. (1995). Artificial linguistic internet computer entity - a.l.i.c.e. https://www.chatbots.org/chatbot/a.l.i.c.e/.

[Young, 1804] Young, T. (1804). I. the bakerian lecture. experiments and calculations relative to physical optics. *Philosophical Transactions of the Royal Society of London*, 94:1–16.