https://repository.uaeh.edu.mx/revistas/index.php/investigium/issue/archive

Boletín Científico INVESTIGIUM de la Escuela Superior de Tizayuca

InvESTigium

ISSN: 2448-4830

Publicación semestral, Vol. 11, No. Especial (2025) 278-284

"Cuarto Congreso Internacional de Investigación e Innovación en Tendencias Globales y Segundo Coloquio de Posgrado en Gestión y Desarrollo de Nuevas Tecnologías"
24 –30 de octubre de 2025

Aplicación Q-Learning en la generación de trayectorias en entornos 2D

Q-Learning application in the generation of Paths in 2D environment

Víctor Tomás Tomás Mariano [a], Jorge Hernández Camacho [b]

Abstract:

This paper presents the development of a GUI to build different search environments (EdB) according to the user's configuration. The environment is random in a grid form with representation in rows and columns. The number of free cells between walls of an environment affects its size, making it more complex to search for a possible solution. Reinforcement learning is applied, specifically the Q-Learning algorithm, to an EdB with multiple free corridors (k> = 3), in which free cells, obstacle or wall cells, and a final cell are identified. A 2D graphical representation of the EdB generated by Q-Learning is made, in which any free cell can be chosen from which to navigate by simply making the movement or color indicated in the cell of the current position, this allows an agent or explorer to reach the final cell. The output of the Q-Learning algorithm is stored in a text file with labels representing the main movements: up, down, right, and left. The 2D graphical representation of the EdB is saved in BMP image format. In the tests conducted, different EdBs were configured, obtaining excellent results with different routes for navigating the environment. Tabular data was generated representing the percentage of movements contained in the EdB and the total number of cells in the environment. The Q-Learning algorithm converges well in extended environments where dimensions range from hundreds to thousands of cells.

Keywords:

Q-Learning, Environment, GUI, path

Resumen:

Este trabajo presenta el desarrollo de una GUI para construir diferentes entornos de búsqueda (EdB) conforme a la configuración del usuario, el entorno se genera aleatoriamente en forma de cuadrícula, representada por filas y columnas. El número de celdas libres entre muros de un entorno afecta el tamaño del mismo lo que lo hace más complejo para buscar una posible solución. Se aplica el aprendizaje por reforzamiento específicamente el algoritmo Q-learning en un EdB con múltiples pasillos libres (k>=3) en la se identifican celdas libres, celdas obstáculo, y una celda final. Se realiza una representación gráfica en 2D del EdB generado por Q-learning en la que se puede elegir cualquier celda libre desde la cual navegar al realizar el movimiento en el color indicado de la celda respecto a la posición actual, esto permite que un agente logre alcanzar la celda final. La salida del algoritmo Q-learning se almacena en un archivo de texto con etiquetas que representan los movimientos principales: arriba, abajo, derecha e izquierda, la representación gráfica 2D del EdB se guarda en formato de imagen bmp. Se configuran diferentes EdB en la que se obtienen excelentes resultados con distintas rutas para navegar por el ambiente. Se muestran datos tabulares: dimensiones, total de celdas, porcentaje de movimientos que contiene el EdB y tiempo de entrenamiento. El algoritmo Q-learning converge de manera adecuada en entornos ampliados en la que las dimensiones van desde cientos a miles de celdas lo que afecta la duración del entrenamiento del algoritmo.

***Palabras Clave**:*

Aprendizaje por refuerzo, Entorno, Q-Learning, Ruta

Introducción

En la actualidad existen varios algoritmos para la planificación de trayectorias en entornos virtuales, estos algoritmos se aplican en dos contextos: entornos conocidos o desconocidos, en el primer caso se parte de la idea de conocer el punto de partida y se conoce también el punto destino, lo que da la posibilidad de hacer una búsqueda informada, el explorador realiza movimientos hacia el objetivo que se quiere alcanzar en el entorno, al respecto se aplican algoritmos de búsqueda heurística; en el segundo caso, se tiene una búsqueda no informada, en esta situación se realiza una búsqueda a ciegas,

[a,b] Universidad Autónoma del Estado de Hidalgo | Escuela Superior de Huejutla | Hidalgo | México, https://orcid.org/0000-0001-6623-860X, Email: victor_tomas@uaeh.edu.mx; https://orcid.org/0000-0001-8647-3332, Email: jorge_hernandez@uaeh.edu.mx

Fecha de recepción: 29/09/2025, Fecha de aceptación: 13/10/2025, Fecha de publicación: 12/12/2025

DOI: https://doi.org/10.29057/est.v11iEspecial.16161

para ello la búsqueda se realiza en profundidad o en amplitud, según sea el caso, estos algoritmos tienen buen funcionamiento en la búsqueda de una trayectoria. En ambos tipos de búsqueda se debe considerar el modelado del entorno, que puede ser tipo cuadricula (*grid*) como una matriz de celdas en la que el explorador se mueve en búsqueda de una celda de interés; y otro, es modelar el entorno como un grafo (no dirigido o dirigido) en la que se debe alcanzar nodos adyacentes hasta encontrar el nodo objetivo. Sin embargo, hay otro tipo de búsqueda dentro de la Inteligencia Artificial, que es utilizar un agente en la que navega por un entorno virtual que interactúa con el ambiente capaz de aprender del entorno y en función de ello encontrar un área de interés, mediante un entrenamiento con un sistema de recompensas que evoluciona hasta encontrar las rutas óptimas, este tipo de estrategia se conoce como aprendizaje por reforzamiento.

Descripción de entorno

Modelar el entorno de búsqueda es una de las etapas más importantes dentro de la búsqueda de trayectorias, en el presente trabajo se utiliza un entorno tipo cuadricula, se representa como una matriz de *n* filas por *m* columnas en la que hay celdas libres por las cuáles navega el explorador; también hay celdas de obstáculos que representan celdas "no admisibles" que se desea no transitar por ellas, ya que representan muros o sitios no deseados dentro del área de búsqueda, es por ello que el agente debe tener cuidado en la navegación para evitar "caer" en ellas. En la Figura 1 se representa un entorno en la que se identifican celdas libres etiquetado con "ceros" y celdas obstáculos etiquetados con "unos".

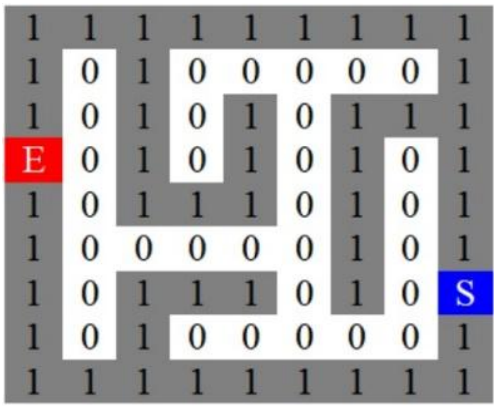

Figura 1. EdB con una solución.

En la Figura 1 se observa un entorno en la que se elige una celda cualquiera ya sea dentro o en un extremo que se identifique como entrada o inicio; también, se elige otra celda cualquiera destino o salida diferente a la celda inicio. En este tipo de entorno se tiene una posible solución entre dos celdas, siempre y cuando las celdas sean accesibles.

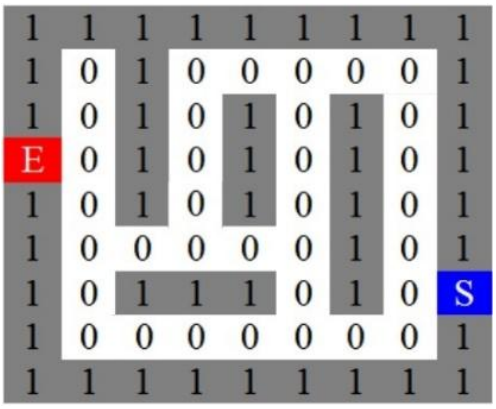

Figura 2. EB más de una solución

En la Figura 2 se muestra un tipo de entorno en la que se tiene una celda como inicio o entrada y otra como final o salida, sin embargo, en este tipo de entornos hay *más de una posible solución* entre dos celdas accesibles, lo que se convierte en un problema para encontrar la ruta más corta entre todas las existentes.

Es posible configurar el entorno a generar, en la Figura 3 se muestra una interfaz para ajustar las características: No. de Filas. No. de Columnas, Cuadro en Pixeles y No. Pasillos que afecta las dimensiones finales del EdB.

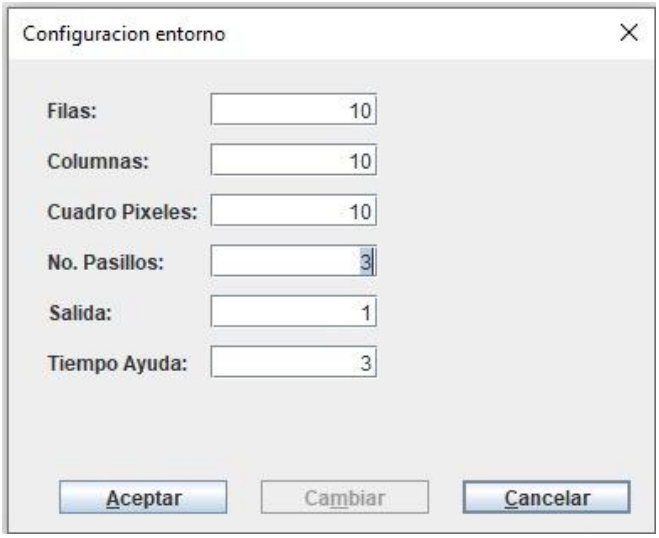

Figura 3. Configuración del entorno, No. de Pasillos se refiere al número de celdas de alto-ancho del pasillo.

Otro tipo de entorno es la que se muestra en la Figura 4, en estos entornos la navegación se puede hacer por varias celdas libres una vez abandonado la celda inicial, la dificultad radica en encontrar las celdas solución. Este tipo de entorno se le puede agregar tantas celdas libres como se desee de acuerdo con la Figura 3, debido a que en el sistema propuesto se configura el número de celdas libres requeridas entre muros.

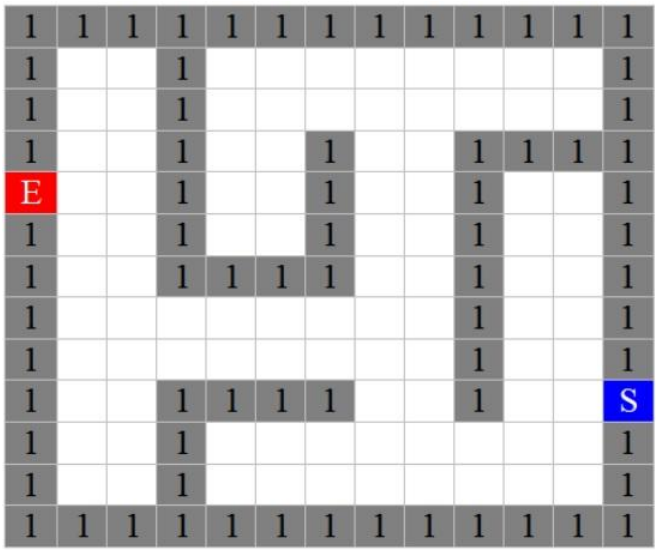

Figura 4. Entorno con 2 celdas libres entre muros.

Para el caso de la Figura 5 tiene tres celdas libres, esta característica hace que el entorno se vuelva más complejo conforme se incremente este valor, también dificulta encontrar la ruta más corta entre la celda inicio y final, debido a que se tienen múltiples trayectorias entre ambas celdas.

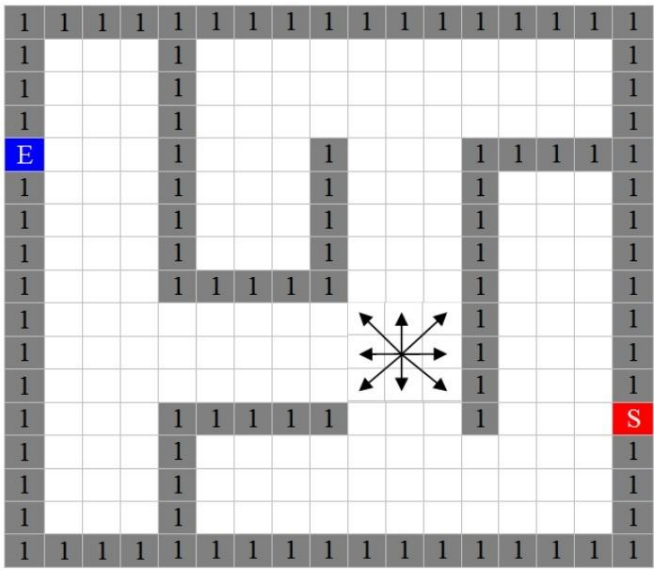

Figura 5. Entorno con 3 celdas libres entre muros.

El sistema propuesto permite crear entornos de búsqueda con varias dimensiones de n filas por m columnas, con k número de celdas libres entre muros de forma cuadrada.

Algoritmos de búsqueda

Para encontrar una trayectoria solución entre la celda inicial y la celda final es necesario modelar el entorno de búsqueda tal como se muestra en la sección anterior, para este tipo de entornos (Figura 1 y Figura 2) existen diversos algoritmos, por ejemplo, *Primera Búsqueda en Profundidad, Primera Búsqueda en Amplitud*, que realizan una búsqueda a ciegas en el entorno, en la mayor parte de su funcionamiento recorren gran parte del ambiente para hallar una solución; también están los algoritmos clásicos como el Dijkstra que maneja un conjunto de pesos o distancias entre las celdas, y con mejores resultados, el algoritmo A* que utiliza una función heurística para decidir hacia cuál celda moverse durante el recorrido del entorno, ambos ofrecen la posibilidad de encontrar una trayectoria más corta entre dos celdas definidas como inicio y fin dentro del ambiente (Quinones-Ramirez, 2023). Los algoritmos mencionados anteriormente son también conocidos como de búsqueda tradicional o clásicos.

En el presente trabajo se utilizan entornos de búsqueda similares en la Figura 5, con diferentes valores de k>=3 que representa la dimensión, correspondiente al número celdas del pasillo, para este tipo de entornos el número de rutas posibles se incrementa, conforme al número de celdas libres entre los muros aumenta. En cada EdB se genera en forma aleatoria una estructura interna única, aunque tengan las mismas dimensiones a diferencia de los entornos usados en la literatura consultada que utilizan EdB con estructura y dimensiones predefinidas o fijas.

Aprendizaje por reforzamiento

El Aprendizaje por Reforzamiento (AR) del acrónimo en inglés (RL) es un área de la IA que tiene varias aplicaciones, una de la cuáles es entrenar una agente que se mueve por un entorno, aprovechando la información del ambiente en forma interactiva, para tomar decisiones en los movimientos que realiza, en estudios recientes podemos hallar diversas aplicaciones: planificación de rutas en vehículos autónomos, drones y plataformas robóticas (Nguyen e tal, (2025); la planificación óptima de tareas y planificación adaptativa en robótica (Wong, 2020); en la navegación autónoma en robots móviles de un entorno desconocido (Quinones, 2023); en la planificación adaptativa de rutas informativas robóticas para entornos desconocidos(Popović e tal, 2024); en el aprendizaje profundo por refuerzo para la planificación de rutas empleando redes neuronales (Yang y Liu, 2024); en los campos potenciales y aprendizaje por refuerzo para entrenar agentes que se adaptan al entorno (Yao *et al*, 2020); en la navegación de un robot en un espacio real y su modelado virtual (Rudkowskyj, 2019) y en el seguimiento de líneas por un robot equipado con cámaras (Delmás, 2025). Estos trabajos dan constancia de la aplicación del AR en la planificación de trayectorias.

El aprendizaje por refuerzo es una rama del aprendizaje automático que tiene diversos tipos de algoritmos según el tipo de entornos en los que se aplique (AlMahamid & Grolinger, 2021). "*El RL permite a un agente aprender en un entorno interactivo mediante prueba y error utilizando retroalimentación*

para maximizar las recompensas acumulativas" (Ramírez y Ramírez, 2023), (olivas e tal, 2023). En la Figura 6 se muestra de manera general la interacción que realiza un agente con el medio ambiente o entorno propuesto, el agente realiza una acción en función de una recompensa obtenida para transitar entre un conjunto de estados.

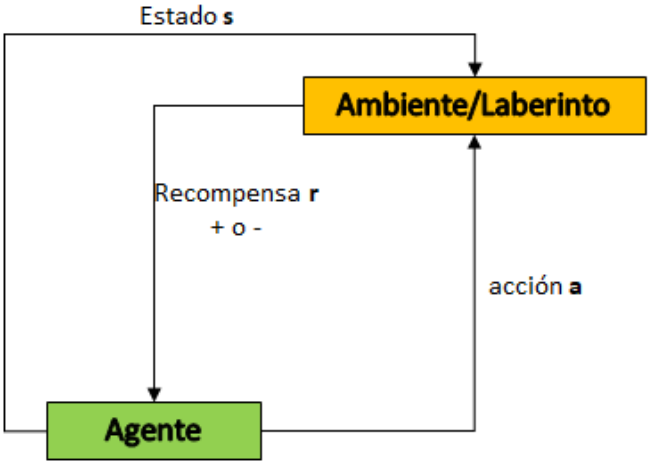

Figura 6. Interacción agente y ambiente.

En el trabajo de Meza, Méndez y Méndez (2023) se muestra una arquitectura general del algoritmo Q-learning en la que se explica como el agente interactúa con el medio ambiente.
Los entornos más utilizados en el aprendizaje por refuerzo son los *grid worlds o métodos tabulares*, por ejemplo, Figura 1 y Figura 2. Estos entornos consisten en una cuadrícula rectangular de celdas, con una celda de inicio y una celda destino. El objetivo es que el agente encuentre la secuencia de acciones que debe realizar (arriba, abajo, izquierda, derecha) para llegar a la casilla objetivo. En varios casos, se añade una celda de "pérdida", "pared" u "obstáculo" que otorga una recompensa o puntos negativos. Al explorar la cuadrícula, realiza diferentes acciones y registra la recompensa (si alcanza la celda destino), el agente puede encontrar una ruta y, cuando la tiene, puede intentar mejorarla para encontrar una ruta ó (rutas), que sean más corta hacia el objetivo.

Algoritmo Q-Learning

Q-learning es un algoritmo de aprendizaje por refuerzo que permite resolver problemas de decisión secuencial en los cuales la utilidad de una acción depende de la secuencia de decisiones, por ejemplo, en la planificación de rutas (Moya, 2024). Utiliza la siguiente fórmula:

$$Q_n(\mathrm{s},\mathrm{a}) = \mathrm{Q}(\mathrm{s},\mathrm{a}) + \alpha * [R(s,a) + \gamma * maxQ(s',a') - Q(s,a)] \quad (1)$$

Donde α es un factor de aprendizaje y γ es una tasa de descuento, estos parámetros son configurados por el diseñador y regulan el aprendizaje, los valores que toman siempre están entre 0 y 1 (Vallejo, 2021). Los valores de la matriz Q, se actualizan con la suma del valor Q anterior y la ecuación que determina la mejor acción en el estado actual. El aprendizaje Q continúa actualizando continuamente el valor Q para cada estado mediante la ecuación (1) (Jang e tal, 2019). La tasa de aprendizaje afecta de manera directa la variación del valor de Q, es decir, cuánto más pequeña sea la tasa de aprendizaje, más lentamente cambia el valor de Q, y viceversa. La tasa de descuento hace que una recompensa positiva se propague en las celdas cercanas o vecinas, esto afecta la velocidad con la que se distribuye una recompensa en el futuro (Caponera, 2021).
Para el presente proyecto se utiliza los siguientes valores $\alpha=0.1$ y $\gamma=0.9$, tal como se recomienda en la literatura consultada.

A continuación, se expone el algoritmo:

Algoritmo Q-learning
Inicializar **Q** con valores aleatorios (por regular a 0)
Para cada episodio repetir:
  Inicializar el estado **s**
  Por cada paso del episodio repetir:
    Según la política (e-greedy) seleccionar la acción **a**
    Ejecutar la acción **a**, observar la recompensa **r** y el nuevo estado **s'**
    Actualizar Q:
$$Q(s,\mathrm{a}) = \mathrm{Q}(s,\mathrm{a}) + \alpha[R(s,a) + \gamma\, max\, Q(s',a') - Q(s,a)]$$
    Actualizar s <-- s'
  Hasta que **s** sea un estado final

Un episodio es un ciclo completo de pasos, es decir, un episodio termina cuando el agente alcanza el estado final, en este caso la celda "F". Tras un nuevo episodio, reinicia el estado del agente, pero con una tabla Q y tabla de recompensas R con valores del episodio anterior, esto es para aprender del resultado previo y llegar a tener una buena política de aprendizaje. En cada episodio el agente recibe del entorno un valor de recompensa positiva o negativa, esto define qué acciones son "buenas" o "malos" (Piña, 2022), similar a transitar por una celda libre o transitar por una celda pared u obstáculo.

El algoritmo Q-learning tiene diversas aplicaciones entre estas se encuentran: juegos, industria, predicciones de mercado, medicina, robótica (Montenegro e tal 2023).

Metodología

En el presente trabajo se crea una GUI para generar entornos de búsqueda con distinto número de celdas libres entre muros, el entorno se visualiza de forma gráfica en un JFrame de Java, esta proporciona facilidades de navegación entre dos celdas marcadas como entrada y salida. El gráfico se guarda en un archivo de texto con símbolos en caracteres como unos ("1") que representan paredes, ceros ("0") que representan celdas libres, una celda "F" que representa la salida del espacio de búsqueda.

El esquema del sistema desarrollado se visualiza en la Figura 7, y se explica de manera general a continuación:

1. Generar un entorno de búsqueda (EdB), configurar las dimensiones en filas, columnas, amplitud en pixeles de cada celda y el número *k* celdas libres deseadas entre muros. El valor de k, afecta la cantidad celdas resultantes del EdB.
2. Visualizar el EdB en forma gráfica en 2D, se manejan eventos para navegar por el EdB desde la entrada a la salida. Al alcanzar la salida el entorno grafico se guarda en un archivo de texto con símbolos o caracteres en "ceros", "unos", una celda "F", así como las filas y columnas del entorno.
3. Se lee el archivo de texto del punto 2 para generar un EdB en forma matricial. El laberinto se almacena en una matriz de n por m, en paralelo se crea la matriz de recompensas R y la matriz Q con valores ceros.
4. Se aplica el algoritmo Q-learning para "resolver" o encontrar las diferentes trayectorias que conectan las celdas del EdB con la celda "F". El resultado con Q-learning se almacena en un archivo de texto con símbolos: "U" arriba, "R" derecha, "I" izquierda y "D" abajo y "X" pared, que hace coincidir con una posición [fila, columna] para facilitar su escritura-lectura.
5. Se realiza la gráfica 2D del EdB con diferentes colores se representan los movimientos hallados por Q-learning. El grafico 2D se almacena como imagen en formato png.

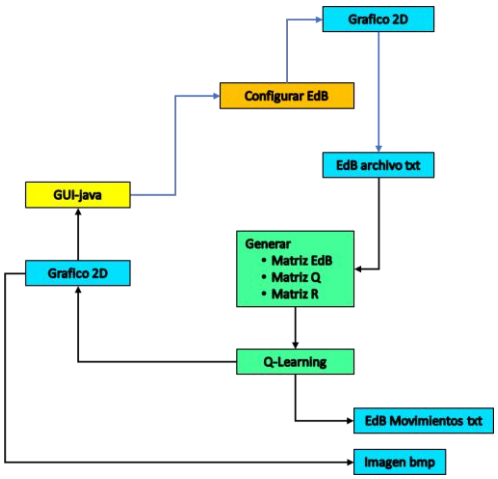

Figura 7. Esquema GUI desarrollado.

El criterio de convergencia del algoritmo Q-learning es realizar un número de episodios tendiente al infinito para hallar Q*, lo que implica que se realicen varias iteraciones para evaluar los resultados obtenidos. En este trabajo se configura 2000 iteraciones por episodio para entrenar al agente lo que arroja resultados satisfactorios en la generación de trayectorias.
Para entornos de búsqueda de cientos de celdas el entrenamiento dura menos de 60 segundos, mientras que para EdB celdas mayores a 3000 se tiene una duración entre 5 y 32 minutos. La duración depende del tamaño del entorno generado.

Resultados

A continuación, se muestran los resultados obtenidos en EdB con diferentes dimensiones. Los EdB se configuran con distintos números de celdas libres entre muros, esta característica hace que en cada prueba realizada se genere un EdB completamente en forma aleatoria. Esta es una aportación de mejora respecto algunos trabajos reportados que modelan q-learning en un entorno fijo o predefinidos.

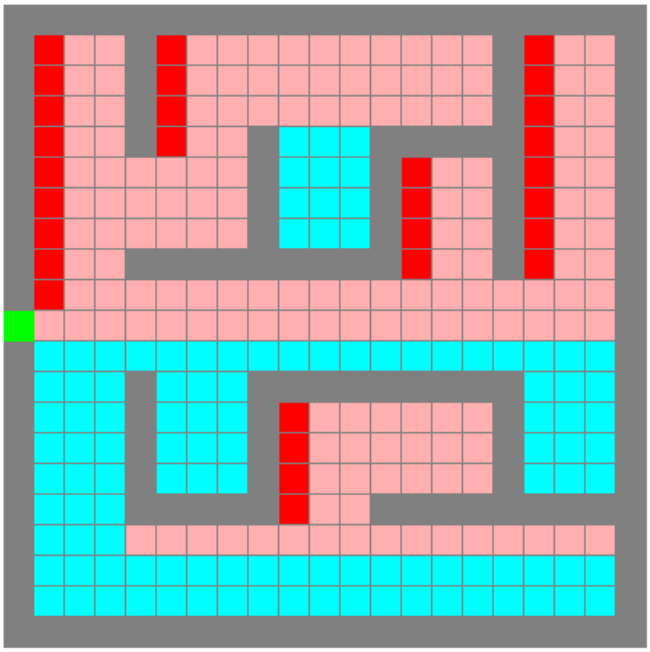

Figura 8. EdB de 21 filas por 21 columnas, (Celeste=Arriba, Rojo=Abajo, Azul=Derecha, Rosa=Izquierda, Verde=Salida).

En la Figura 8 se utilizó un laberinto ampliado de 441 celdas. Cada celda del laberinto corresponde a un estado del ambiente virtual. En cada estado o celda se pueden realizar cuatro movimientos: Derecha, Izquierda, Arriba, Abajo, las cuales causan que el agente se mueva a la celda vecina correspondiente.
Todas las transiciones libres tienen una recompensa de 10, las paredes una negativa de -10, excepto la transición que conduce a un estado terminal -indicado en verde en la Figura 8- la cual da 20 unidades de recompensa.
En la Figura 8, se marca en color verde la celda final "F", en color gris-oscuro las celdas que representan paredes o celdas bloqueadas. Cada celda en color Celeste permite un movimiento hacia arriba; cada celda en color Rojo permite un

movimiento hacia abajo; la celda en color Rosa permite un movimiento hacia la izquierda. En este caso en particular no aparecen celdas con movimiento a la derecha.

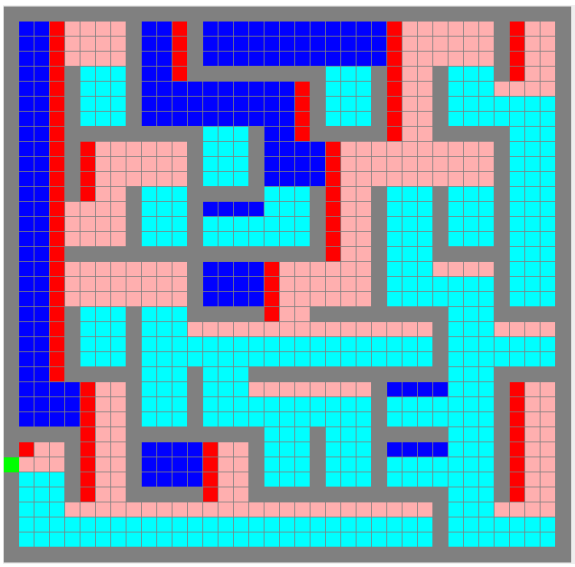

Figura 9. EdB de 37 filas por 37 columnas, (Celeste=Arriba, Rojo=Abajo, Azul=Derecha, Rosa=Izquierda, Verde=Salida).

En la Figura 9, se utilizó un entorno de búsqueda de 1369 celdas, se marca en color verde la celda final "F", en color gris-oscuro las celdas pared. Las celdas en color Celeste permiten movimientos hacia arriba; las celdas en color Rojo permiten movimientos hacia abajo; las celdas en color Rosa permiten movimientos a la izquierda; las celdas en color Azul permiten movimientos hacia la derecha. En este caso en particular se representan los cuatro movimientos para lograr alcanzar la celda final.

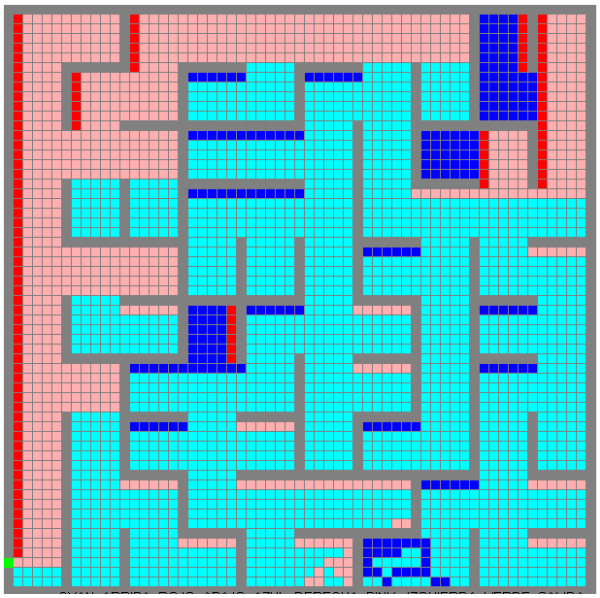

Figura 10. EdB de 61 filas por 61 columnas, (Celeste=Arriba, Rojo=Abajo, Azul=Derecha, Rosa=Izquierda, Verde=Salida).

El sistema genera diferentes entornos de búsqueda, en la que el algoritmo q-learning etiqueta de manera correcta las celdas del entorno. En la Figura 10 y Figura 11 se visualizan dos ambientes con celdas libres k=5 y k = 7, respectivamente. Se observa que la navegación se puede hacer desde cualquier celda libre elegida como origen hasta llegar a la celda final "F", este etiquetado en colores (colorímetro) hace que un explorador navegue por el entorno sin hacer colisiones o choques con las celdas obstáculo que existen, lo hace seguro para alcanzar la meta dentro del entorno.

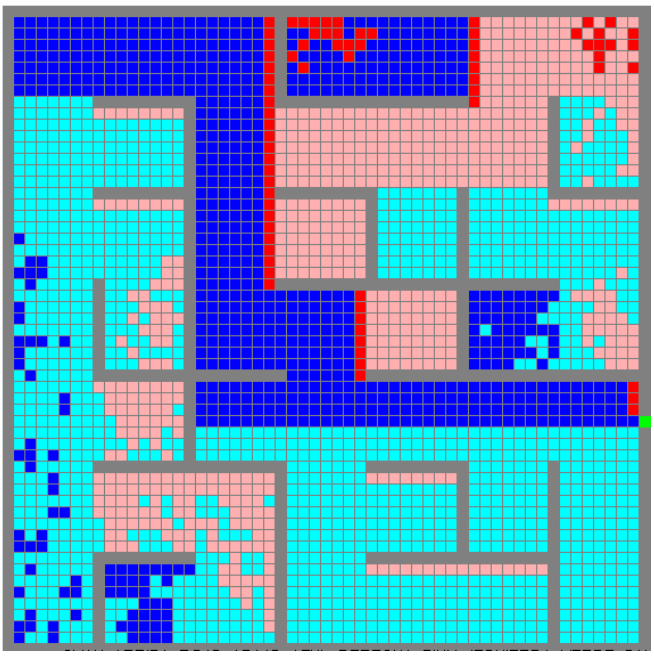

Figura 11. EdB de 57 filas por 57 columnas, (Celeste=Arriba, Rojo=Abajo, Azul=Derecha, Rosa=Izquierda, Verde=Salida).

Tabla 1. Diferentes entornos de búsqueda.

Dimensiones		Total Celdas	Valor k	Tiempo	Tipo de Movimientos (%)			
Fil	Col				Izq.	Der.	Arriba	Abajo
21	21	441	3	00:00:28	35.60	0.0	25.17	6.57
37	37	1369	3	00:01:22	19.94	13.44	31.48	5.91
61	61	3721	5	00:03:32	23.73	6.23	47.72	2.79
61	61	3721	5	00:03:36	22.60	5.94	49.15	2.80
57	57	3249	7	00:07:38	19.45	22.49	40.1	2.21
91	91	8281	5	00:32:11	23.29	16.15	32.32	9.67

Se realizaron varias pruebas con diferentes dimensiones, en la tabla 1 se muestran datos obtenidos respecto al total de celdas, valor k, porcentaje de movimientos y el tiempo de entrenamiento que se obtuvo con el algoritmo Q-learning. Se resalta que a mayor número de celdas el tiempo más largo fue de 32 minutos y 11 segundos en encontrar las trayectorias del EdB.

Conclusiones

El algoritmo Q-learning para entornos de búsqueda tipo grid ofrece buenos resultados, en pruebas con EdB de diferentes dimensiones se obtuvieron trayectorias que conducen a la salida. Algo que se observa en la ejecución del algoritmo es el tiempo que emplea en encontrar una tabla Q, que influye por el número de celdas, sin embargo, la matriz de movimientos final proporciona trayectorias para hallar la salida desde cualquier celda libre elegida dentro del entorno. Para fines prácticos el EdB y los movimientos asociados se almacenan en archivos de texto lo que brinda la posibilidad de volver a graficar el EdB para una posterior evaluación.

Como trabajo futuro, agregar movimientos en diagonal, así mismo un temporizador para visualizar la navegación del agente de forma gráfica el recorrido que realiza desde cualquier celda elegida, encontrar el número más eficiente del cómputo del número de ciclos mínimo, ya sea este estático o dinámico en base a los resultados obtenidos.

Referencias

[1] AlMahamid, F., & Grolinger, K. (2021, September). Reinforcement learning algorithms: An overview and classification. In 2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) (pp. 1-7). IEEE. https://doi.org/10.48550/arXiv.2209.14940

[2] Caponera R. R. (2021). Comparación de algoritmos de aprendizaje por refuerzo basados en Q-Learning., https://hdl.handle.net/10630/23354

[3] Delmás, E. L. (2025). Seguimiento de líneas con técnicas de aprendizaje por refuerzo en robótica móvil, [Trabajo fin de grado en Ingeniería en informática, Universidad Politécnica de Madrid], https://oa.upm.es/90129/1/TFG_EDUARDO_LOPEZ_DELMAS.pdf

[4] Jang B., Kim M., Harerimana G. and Kim J. W., "Q-Learning Algorithms: A Comprehensive Classification and Applications," in *IEEE Access*, vol. 7, pp. 133653-133667, 2019, doi: 10.1109/ACCESS.2019.2941229.

[5] Moya Quinatoa, K. A. (2024). Aplicación del algoritmo de aprendizaje por refuerzo Q-Learning para la generación de trayectorias óptimas en plataformas robóticas, [Trabajo de titulación previo al título en Ingeniero en Tecnologías de la Información, Universidad Técnica de Ambato], https://repositorio.uta.edu.ec/handle/123456789/42440

[6] Montenegro Meza, M. A., Menchaca Méndez, R., & Menchaca Méndez, R. (2023). Una Introducción amable pero riguroza al aprendizaje por refuerzo. ReCIBE. Revista electrónica de Computación, Informática, Biomédica y Electrónica, 12(1), 1-15. https://doi.org/10.32870/recibe.v12i1.268

[7] Nguyen, T. T., Nahavandi, S., Razzak, I., Nguyen, D., Pham, N. T., & Nguyen, Q. V. H. (2025). The Emergence of Deep Reinforcement Learning for Path Planning. https://doi.org/10.48550/arXiv.2507.15469

[8]Olivas, E. S., Isla, M. A. S. M., Cruz, R. G., & Caballer, B. C. (2023). Sistemas de aprendizaje automático. Editorial: RA-MA, ISBN: 9788419444981,

[9] Pina Navarro, M. (2022). Aplicación de técnicas de aprendizaje por refuerzo a navegación visual, [Trabajo fin de Máster, Universidad de Alicante], http://hdl.handle.net/10045/124262

[10] Popović, M., Ott, J., Rückin, J., & Kochenderfer, M. J. (2024). Learning-based methods for adaptive informative path planning. Robotics and Autonomous Systems, 179, 104727. https://doi.org/10.48550/arXiv.2404.06940

[11] Quinones-Ramirez, M., Rios-Martinez, J., & Uc-Cetina, V. (2023). Robot path planning using deep reinforcement learning. https://doi.org/10.48550/arXiv.2302.09120

[12] Ramírez, C., & Ramírez, W. (2023). Programación de inteligencia artificial: Curso práctico. Editorial: RA-MA. ISBN: 9788419857019, https://uaeh.bibliotecasdigitales.com/read/9788419857019/index

[13] Rudkowskyj H. S. (2019). *Aprendizaje por refuerzo en sistemas robóticos*. [Trabajo Fin de Grado en ingeniería en Tecnologías Industriales, Universidad Politécnica de Madrid] , https://oa.upm.es/56678/

[14] Vallejo Del Moral, M. (2021). Algoritmo Deep Q-Learning para el aprendizaje por refuerzo de una estrategia de conducción en 2D. https://academica-e.unavarra.es/handle/2454/40521

[15] Wong, C. (2020). Adaptive task planning and motion planning for robots in dynamic environments. doi: 10.48730/1r89-y325

[16] Yang K. and Liu L., (2024), "An Improved Deep Reinforcement Learning Algorithm for Path Planning in Unmanned Driving," in *IEEE Access*, vol. 12, pp. 67935-67944, doi: 10.1109/ACCESS.2024.3400159.

[17] Yao Q. *et al*., (2000), "Path Planning Method With Improved Artificial Potential Field—A Reinforcement Learning Perspective," in *IEEE Access*, vol. 8, pp. 135513-135523, doi: 10.1109/ACCESS.2020.3011211.