Ciencia Latina Revista Científica Multidisciplinar
Julio-Agosto, 2023, Volumen 7, Número 4.
https://doi.org/10.37811/cl_rcm.v7i4.7440

Propuesta para Encontrar una ruta más Corta en un Entorno de Búsqueda 2D

Victor Tomás Tomás Mariano[1]
victor_tomas@uaeh.edu.mx
https://orcid.org/0000-0001-6623-860X
Universidad Autónoma del Estado de Hidalgo
Escuela Superior de Huejutla,
Huejutla, Hidalgo, México.

Jorge Hernández Camacho
jorge_hernandez@uaeh.edu.mx
https://orcid.org/0000-0001-8647-3332
Universidad Autónoma del Estado de Hidalgo
Escuela Superior de Huejutla,
Huejutla, Hidalgo, México.

Felipe de Jesús Núñez Cárdenas
felipe_nunez@uaeh.edu.mx
https://orcid.org/0000-0002-2462-3654
Universidad Autónoma del Estado de Hidalgo
Escuela Superior de Huejutla,
Huejutla, Hidalgo, México.

RESUMEN

Para construir espacios de búsqueda (Laberintos) existen diversos algoritmos, estos se clasifican en laberintos simplemente conectados (LSC) y laberintos de conexión múltiple (LCM), los LSC tienen una única solución para dos celdas indicadas como entrada y salida. También hay diversos algoritmos para buscar la solución de un laberinto [1], [2], [8], Dijkstra es uno de los más utilizados en la representación con grafos, otros algoritmos, utilizan una representación matricial, con el uso de métricas de distancia utilizan información a priori para alcanzar el objetivo o meta, ejemplo de ello es A* o D* [2, 3]. En este trabajo se propone una estrategia para encontrar la trayectoria en un entorno de búsqueda o laberinto con una representación matricial tipo LSC, en la que se aprovecha información a priori que se obtiene durante *la construcción* del espacio de búsqueda, los movimientos se representan en forma de etiquetas numéricas que facilitan la navegación por una ruta en el entorno; en la ruta encontrada se identifican *patrones binarios* para realizar movimientos en diagonal de esa manera obtener una trayectoria más corta entre dos celdas.

Palabras clave: *búsqueda; laberinto; trayectoria.*

[1] Autor principal.
Correspondencia: victor_tomas@uaeh.edu.mx

Proposal to find a Shortest Path in a 2D Search Environment

ABSTRACT

To build search spaces (Mazes) there are various algorithms, these are classified into simply connected mazes (LSC) and multiple connection mazes (LCM), LSCs have a single solution for two cells indicated as input and output. There are also various algorithms to find the solution of a maze [1], [2], [8], Dijkstra is one of the most used in graph representation, other algorithms use a matrix representation, with the use of distance metrics they use a priori information to achieve the objective or goal, an example of this is A* or D* [2], [3]. In this work, a strategy is proposed to find the trajectory in a search or maze environment with an LSC-type matrix representation, in which a priori information obtained during the construction of the search space is used, the movements are represented in the form number labels, which facilitate navigation in the environment. In the path found, binary patterns are identified to perform diagonal movements in this way to obtain a shortest trajectory between two cells.

***Keywords*:** *search; maze, path*

Artículo recibido 20 julio 2023

Aceptado para publicación: 20 agosto 2023

INTRODUCCIÓN

La planeación o planificación de trayectorias es una de las áreas de la inteligencia artificial que ha tenido mucho auge en los últimos años, su modelado va desde el manejo de agentes inteligentes (robots) que interactúan en su medio ambiente para la toma de decisiones, hasta el de entornos virtuales que utiliza paradigmas para representar un entorno de búsqueda, entre estos, está la teoría de grafos, y rejilla(grid) de tipo rectangular con dimensiones de *n por m* en la que se definen un conjunto de estados deseados para navegar en su interior; la idea principal es que dado dos puntos origen y destino respectivamente, el problema es encontrar la solución o trayectoria que conecte dos puntos elegidos; en ese sentido, hay diferentes situaciones o casos para iniciar una búsqueda, la primera se conoce como *búsqueda no informada o a ciegas*, en este caso los algoritmos de primera búsqueda en amplitud, primera búsqueda en profundidad y el algoritmo de Lee-Moore son de utilidad; la segunda estrategia es hacer una *búsqueda informada*, el algoritmo A* y sus variantes de los aplicados; si el entorno se representa mediante grafos *no dirigidos o dirigidos* según sea el caso del entorno virtual, entonces se aplican algoritmos para encontrar la ruta más corta entre dos nodos, uno de los más aplicados es el de *Dijkstra* [4, 8] que maneja valores positivos en sus aristas; el algoritmo A* se puede aplicar para grafos o entornos tipo rejilla en la solución de entornos de búsqueda [2], [3], [8].

Hay algoritmos que se enfocan únicamente en *construir entornos* de búsqueda, en este trabajo se analizan dos principales: algoritmo de Kruscal y Algoritmo de Prim, de los cuáles se aprovecha información a priori que se obtiene en el proceso de *construcción* para proponer una estrategia para encontrar la solución o trayectoria que une dos celdas cualesquiera dentro de un laberinto, también, se encuentra una primera solución entre dos celdas que utiliza etiquetas numéricas con movimiento en horizontal o vertical, en un segundo momento, se mejora la solución para identificar puntos de interés en los que se identifican patrones de celdas binarias para eliminar celdas, y agregar movimientos en diagonal, lo que hace que la trayectoria se reduzca y por lo tanto sea más corta.

METODOLOGÍA

Hay varios trabajos en la literatura que tratan sobre la búsqueda de solución [14] de un laberinto, algunos problemas utilizan búsqueda informada, otros, la no informada; en ambos casos analizan la solución de un laberinto ya construido a partir del cual se elige un punto origen, el objetivo es lograr alcanzar el

punto destino, y es en ese proceso de encontrar la solución se aplican distintos algoritmos. Ahora bien, encontrar la solución no es lo mismo que construir un entorno de búsqueda, para hacerlo también existen varios algoritmos, los más significativos, sus fases de desarrollo y esquema de funcionamiento se analizan a profundidad en el trabajo [1], [2], [5].

Encontrar la solución de un laberinto depende de la representación del espacio de búsqueda, para el trabajo que aquí se presenta es para entornos tipo rejilla (grid) rectangular; sin embargo, este tipo de representación también se puede combinar con un grafo, en la que el problema entonces se convierte en encontrar la ruta entre dos nodos [9]-[11].

En [15] "*se presentan las técnicas de búsqueda en amplitud y en profundidad, las cuales son técnicas de inteligencia artificial, para la solución de un problema de gran complejidad matemática como lo es la solución de un laberinto de estructura desconocida*", los algoritmos mencionados *DFS: Primera Búsqueda en Profundidad (Deep Search Problem) y BFS: Primera Búsqueda en Amplitud (Breadth First Problem)* son algoritmos que hacen una búsqueda a ciegas que encuentran la solución a un laberinto si es que existe, uno de ellos es más eficiente que el otro, pero en su búsqueda recorren gran parte del área del entorno, ambos algoritmos se puede usar en problemas tipo rejilla o con grafos.

En trabajo citado [5], se analizan los algoritmos para construir laberintos: Aldous-Brother, Backtrack; también se analiza la solución de laberintos con el algoritmo Dijkstra y el algoritmo búsqueda profunda, hace un comparativo para encontrar la solución de un laberinto con dimensiones parecidas. Los laberintos se representan en entornos tipo rejilla con dimensiones de n por m.

En trabajo con la citación [11], se trata el problema de "*búsqueda de una ruta óptima en un entorno virtual, caso de estudio laberinto ampliado, mediante la implementación del algoritmo A*(A Estrella), el cual utiliza dos listas que controlan la secuencia de exploración en el laberinto ampliado y de una medida heurística que lo convierte en un método rápido y óptimo*". En este trabajo en particular se utiliza la búsqueda informada debido a que desde un inicio se sabe la celda que se desea alcanzar, el problema es tomar decisiones que acerquen al objetivo perseguido, para ello hace uso de métricas de distancia, también conocidas con heurísticas, los laberintos son tipo rejilla.

La robótica es un área que utiliza algoritmos [1], [4] para la toma de decisiones para que un robot encuentre la trayectoria en un entorno de búsqueda. En el trabajo citado [1] se discuten y analizan

estrategias para resolver un laberinto, los algoritmos se clasifican en dos categorías, entornos conocidos: algoritmo de Lee, algoritmo Dead-end, algoritmo Soukup, algoritmo Maze-router, algoritmo Hadlock, Algoritmo Flood-Fill y algoritmo de búsqueda Heurística; entornos desconocidos, algoritmo Wall-follower, algoritmo Tremaux, algoritmo Random-Mouse y Algoritmo Pledge, para cada algoritmo se menciona su funcionamiento general y una descripción de las fases o etapas que contempla. En este otro trabajo citado [10] se analizan algoritmos para resolver laberintos en 2D como lo son Random Mouse and Wall Follower.

En el presente artículo se analizan dos algoritmos principales para construir espacios de búsqueda –laberintos- y se aprovecha la información que se obtiene en el proceso de construcción, hacer esto permite almacenar en una lista las posiciones [i, j] de los muros "excavados" o apertura de muros conforme se avanza en la construcción. También se propone una estrategia para encontrar una solución entre dos celdas cualesquiera.

Búsqueda basada en Heuristicas

Algoritmo A*.

Este algoritmo es uno de los más aplicados en la búsqueda de rutas, tiene la característica de encontrar el camino más corto en un grafo o en entornos de búsqueda tipo mallado o grid [1], [6], [7], [8]. Realiza una búsqueda informada, debido a que inicialmente se elige un punto o nodo origen, similarmente se elige otro nodo en el entorno como destino u objetivo, es decir, se sabe a dónde se desea ir, pero no se sabe el trayecto a seguir para lograr conseguir una ruta. Este algoritmo hace uso de una heurística para suponer una distancia que falta por recorrer desde el nodo actual hasta el nodo destino, utiliza las siguientes premisas: Una función g(n) que representa el costo de ir desde el nodo origen al nodo actual, h(n) que es el costo que falta por recorrer del nodo actual al nodo destino, f(n)= g(n) + h(n) que es el costo menor elegido durante la elección del nodo o celda; dependiendo del entorno utiliza una heurística, hace uso de una métrica de distancia como la euclidiana o manhattan en entornos tipo rejilla debido a que se representa en una matriz rectangular, aunque en la literatura se pueden encontrar otro tipo de heurística a utilizar, depende de la manera en que se represente el entorno de búsqueda. Para un análisis del funcionamiento del algoritmo A* se recomienda consultar [1], [4], [9] que se aplica en entornos tipo rejilla.

Pseudocódigo A*.

Para la entrada del laberinto [2]: calcular *f(n), g(n) y h(n*), e insertar a Lista Abierta (LA).

Repetir mientras LA contenga elementos

I. Extraer en LA la celda con menor costo f(n) (llamémosle celda analizada), y se inserta a Lista Cerrada (LC).

II. Para cada celda vecina transitable a la celda analizada.

    1. Si celda vecina no se encuentra en LC ni en LA, hacer que su celda padre sea la celda analizada, calcular los costos f(n), g(n) y h(n), e insertarla a LA. Si sucede que celda vecina es la salida del laberinto, el camino se encuentra al retroceder en dirección de los padres hasta alcanzar la celda de la que inicialmente se partió, y el proceso de búsqueda termina.
    2. En otro caso, si ya se encuentra en LA, verificar si el trayecto a través de celda analizada es menor que el que ya tiene en su costo g(n). Si es así, cambiar su padre a celda actual y recalcular sus costos g(n) y f(n).
    3. En caso de que LA se encuentre vacía y no se ha llegado a la salida del laberinto, entonces no existe camino posible.

Resultado de aplicar el algoritmo A* en un entorno de búsqueda tipo rectangular o grid se recomienda consultar [2], [3].

Existen variantes del algoritmo A*, por ejemplo, el algoritmo D* que se utiliza para entornos dinámicos [2].

ALGORITMO DE CONSTRUCCIÓN ESPACIOS DE BÚSQUEDA.

La construcción al azar se basa en el enfoque de cavar túneles. Como su nombre lo indica, se refiere a cavar túneles a lo largo y ancho del laberinto hasta cubrirlo en su totalidad, como la explotación de una mina [3], [12].

Algoritmo de Construcción Kruskal.

Este algoritmo genera Laberintos de Conexión Simple. Cava túneles en varias secciones del laberinto, como se podrá observar a continuación:

1. Especificar el tamaño o las dimensiones del laberinto de acuerdo con las habitaciones que se desea que contenga.

2. Dividir el laberinto en habitaciones en base a sus dimensiones y etiquetarlas con una identificación única.
3. Seleccionar una habitación al azar y cavar un túnel hacia una habitación adyacente (en caso de existir más de una, elegir al azar) sólo si tiene diferente etiqueta, y etiquetar la habitación o habitaciones que se unieron con la identificación de la habitación inicial; repitiendo este proceso hasta que todas las habitaciones tengan la misma etiqueta.
4. Marcar la ubicación de la entrada o del punto inicial de la exploración y la ubicación de la salida.

En el trabajo citado [12] se ejemplifica está técnica paso a paso.

Algoritmo de Construcción Prim's.

Este algoritmo genera laberintos simplemente conectados, el procedimiento es el siguiente.

Considera dos tipos de habitaciones:

Habitaciones etiquetadas con I son aquellas que forman parte del laberinto y han sido cavadas.

Habitaciones etiquetadas con F que no han sido cavadas, pero que son adyacentes a una habitación I.

1. Especificar el tamaño o las dimensiones del laberinto de acuerdo con las habitaciones que se desea que contenga.
2. Dividir el laberinto en habitaciones en base a sus dimensiones y etiquetarlas con una identificación única.
3. Seleccionar una habitación inicial, etiquetarla como I y etiquete sus habitaciones adyacentes con F.
4. Seleccione una habitación F aleatoriamente. Cavar un muro de la habitación I a la habitación F; cámbielo a I y cambie sus habitaciones adyacentes que no son I a F.
5. Repita este proceso mientras haya habitación F.
6. Marcar la ubicación de la entrada o del punto inicial de la exploración y la ubicación de la salida.

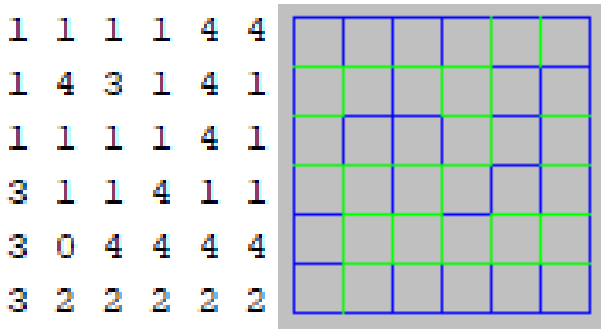
En [12] se ejemplifica está técnica paso a paso.

DISCUSIÓN

Solución de Laberintos con Etiquetado de Movimientos (SoLEM).

Para encontrar la solución en un entorno de búsqueda virtual, se propone usar información a priori que se obtiene al cavar un muro en el laberinto, de tal manera que, durante la elección del muro, se asigna uno de 4 posibles etiquetas numéricas que simbolizan un movimiento desde la celda de dónde se parte a la celda que se cava el muro (paso 3 en algoritmo de Kruscal, paso 4 en algoritmo de Prim. En la figura 1 enciso a), los valores numéricos representan respecto a la celda actual: el "0" es la celda origen dónde se inicia la construcción del laberinto es la celda inicial, el "1" representa un movimiento hacia abajo, el "2" un movimiento hacia arriba, el "3" un movimiento a la derecha, el "4" movimiento a la izquierda; esto significa que, desde una celda diferente a la "0", se puede "alcanzar" la celda origen al hacer el movimiento indicado en la celda respectiva.

Figura 1. *Matriz con etiquetas numéricas y representación gráfica 2D del entorno.*

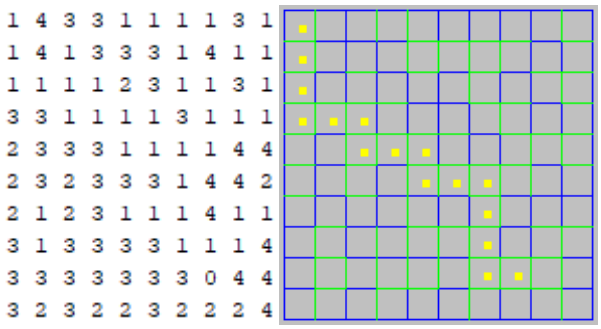


a) b)

En la figura 1 enciso b), se muestra la representación gráfica del laberinto, en el entorno 2D las líneas azules, que representan los muros del laberinto, mientras que las líneas verdes representan los muros "cavados" o accesos libres para navegar por el espacio de búsqueda. La propuesta de etiquetado numérico de celdas, tiene la ventaja que cualquier celda distinta a la "0" se puede elegir como destino, en el proceso de buscar la ruta o trayectoria, por ejemplo, desde la esquina superior izquierda, bastaría con 1.- leer la etiqueta de la celda [i, j], 2.- ejecutar el movimiento, 3.- mientras la celda [i, j] sea diferente de cero, repetir pasos 1 y 2, eventualmente las celdas visitadas lo llevaran a la celda "0" o inicial, ver figura 2. Se

puede observar que, en el proceso de búsqueda de la ruta, el explorador se mueve directamente a la celda de interés, es decir, la propuesta no hace una búsqueda a ciegas, tal como lo hacen los algoritmos primera búsqueda en profundidad o primera búsqueda en amplitud, evita buscar en otras partes del laberinto, lo que lo hace más eficiente. El conjunto de celdas que conforman la ruta *desde la esquina superior izquierda a la celda inicial*, tiene movimientos en vertical u horizontal, según sea el caso, la representación de puntos se muestran en color amarillo en la figura 2.

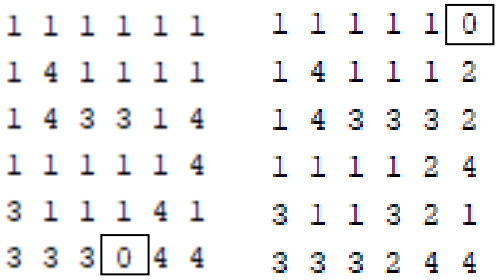
Figura 2. *Matriz numérica asociada con representación gráfica y trayectoria.*



Cambiar la celda inicial.

La sección anterior supone que siempre se elige la celda "cero" como celda origen, sin embargo, esta premisa no siempre debe ser así, a fin de representar un cambio, se elige otro punto, por ejemplo, la celda superior derecha, aunque se puede elegir cualquier otra celda, este ajuste lo hace dinámico, al menos en los puntos origen y destino dentro del mismo entorno virtual de búsqueda, para ello se designa una nueva "celda inicial", y se identifican las celdas que actualizan su valor numérico; estos nuevos valores también permiten "alcanzar" a la nueva celda "0". En la figura 3 se muestra los cambios numéricos realizados.

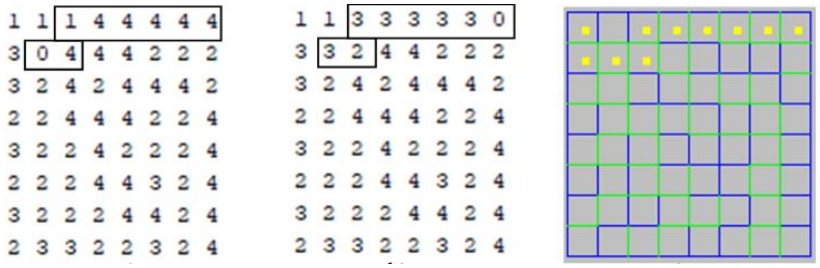
Figura 3. Cambio de celda origen "0".



a) b)

En la figura 3, en a) la celda “0”, posición (6,4), se designa de forma automática como resultado de aplicar el algoritmo de construcción, en b) se ha designado otra celda origen “0” definido por el usuario, posición (1,6). Se observa que se ajustan las etiquetas numéricas a fin de que ahora las celdas llevan a la nueva celda “0”

Figura 4. *Ajuste de celda origen. a) Celda “0” inicial, b) Celda “0” definido por el usuario, c) Representación gráfica.*



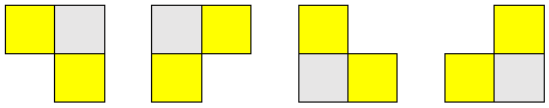
a) b) c)

Por ejemplo, en figura 4 enciso a) la celda “0” se elige como resultado de aplicar el algoritmo de construcción, recordemos que esta elección es aleatoria, nuevas etiquetas numéricas aparecen en enciso b), en enciso c) se visualiza la representación gráfica y trazo de la nueva la ruta.

Estrategia para mejorar ruta.

En las secciones previas, la trayectoria entre la celda destino y la celda origen, contempla movimientos en horizontal o vertical, según sea el caso, sin embargo, se mejora el trayecto del camino al identificar puntos de interés dentro de la misma ruta, para ello se usan los siguientes patrones en el conjunto de puntos que forman la solución.

Figura 5. Celdas patrón para identificar un movimiento en diagonal



En la figura 5 se muestran los 4 casos posibles que se equiparan en el conjunto de puntos que conforman una ruta, las celdas sombreadas en gris con vecinos en diagonal en color amarillo; las celdas grises permiten hacer un movimiento en diagonal entre las celdas amarillas. Al eliminar las celdas en color gris se reduce la longitud de la trayectoria, de esta forma se realiza un número menor de movimientos para ir de la celda origen a la celda destino. Esta estrategia de eliminar movimientos en horizontal o

vertical hace que se obtenga un camino más corto. En la figura 6 se muestra una solución más corta con movimientos de celdas en horizontal, vertical o diagonal, según sea el caso, se representan en color rojo la trayectoria más corta entre las dos celdas.

Figura 6: *Ruta más corta, celdas con puntos rojos.*

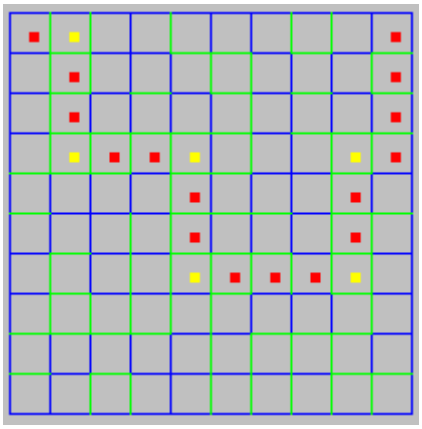
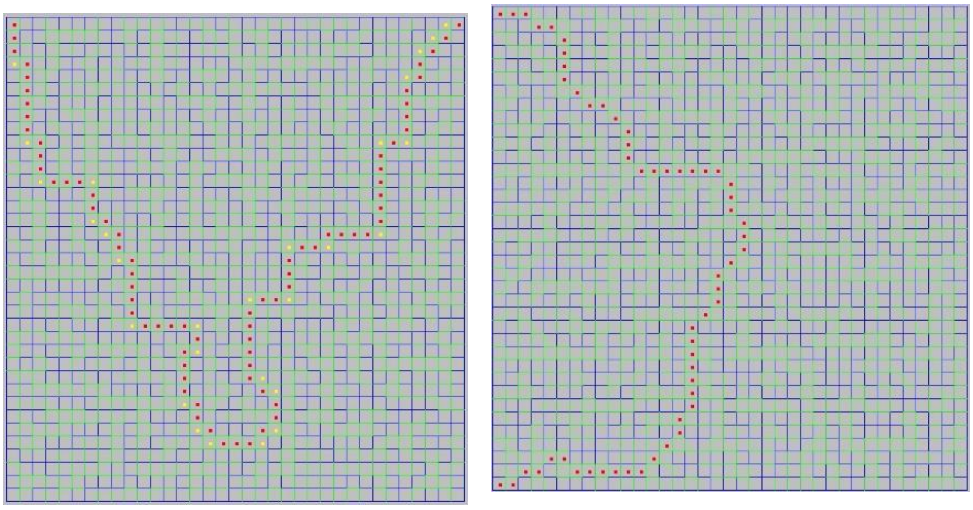


Figura 7. *Laberintos con diferentes rutas.*



a) b)

En la figura 7, se muestran resultados obtenidos en entornos de búsqueda de 37 filas por 37 columnas, en enciso a) la celda destino en la parte superior izquierda, la celda origen en la celda esquina superior derecha, los puntos amarillos son discriminados para obtener una ruta más corta, en enciso b) se muestra, la ruta con la propuesta planteada, se muestra la ruta más corta con puntos en color rojo entre las celdas origen y destino, respectivamente.

TRABAJO FUTURO

Se puede observar que la propuesta genera resultados satisfactorios al encontrar la solución entre dos celdas cualesquiera dentro del entorno de búsqueda, la ubicación de celdas para realizar movimientos en diagonal es una mejora para reducir la longitud del camino. Existen algoritmos que representan un entorno de búsqueda con grafos u otras que trabajan en entorno tipo rejilla (grid) con la utilización de heurísticas para encontrar una solución entre dos celdas que obtienen rutas cortas, Dijkstra y A*, por mencionar solo 2 algoritmos, sin embargo, en este trabajo de investigación se propone una nueva estrategia para encontrar una ruta mejorada entre dos celdas en un laberinto.

Una mejora posible es extender la propuesta para aplicarse en entornos tipo rejilla en la que exista dos o más caminos que unan a dos puntos cualesquiera, solucionar laberintos de conexión múltiple (LCM).

REFERENCIAS

[1] Alamri, S., Alshehri, S., Alshehri, W., Alamri, H., Alaklabi, A., & Alhmiedat, T. (2021). Autonomous maze solving robotics: Algorithms and systems. *International Journal of Mechanical Engineering and Robotics Research*, 50-62.

[2] Balderrama-Garcia, C. A., & Orozco-Rojas, U. P. (2021). Implementación de un algoritmo D* modificado para planificación de trayectorias. *Investigación Básica y Aplicada, 8*(16), 2026-232. Obtenido de https://repositorio.cetys.mx/handle/60000/1299

[3] Bu, Z., & Korf, R. E. (2022). A*+BFHS: A Hybrid Heuristic Search Algorithm. *In Proceedings of the AAAI Conference on Artificial Intelligence, 36*(9), 10138-10145. doi: https://doi.org/10.1609/aaai.v36i9.21253

[4] Castro, M., & Arturo, C. (23 de 09 de 2021). Implementación de planificadores de trayectorias Dijkstra y Astra para un robot móvil diferencial. Colombia, Colombia, Colombia. Obtenido de http://hdl.handle.net/11634/35769

[5] Cruz-Ruiz, I. O., Lara-Velázquez, P., De-Los-Cobos-Silva, S. G., Rincón-García, E. A., Mora-Gutiérrez, R. A., & Gutiérrez-Andrade, M. A. (2019). Un algoritmo estocástico para resolver laberintos. *Revista de Matemática: Teoría y Aplicaciones,* 319-338.

[6] Fernandez, R., & Monserrat, A. (20 de 05 de 2019). *Evaluación del algoritmo Theta* para planeación de trayectorias.* Obtenido de Centro Nacional de Investigación y Desarrollo Tecnológico, Repositorio tesis: https://rinacional.tecnm.mx/jspui/handle/TecNM/5452

[7] Hernández Hernández, G.; Tomás Mariano, V. T. (2012). Búsqueda de Rutas Implementando A* en un Entorno Virtual 2D. Escuela Superior de Huejutla. UAEH. México

[8] Jamal, A., Tauhidur-Rahaman, M., Al-Almadi, H. M., Ullah, I., & Zahid, M. (2020). Intelligent Intersection Control for Delay Optimization: Using Meta-Heuristic Search Algorithms. *12*(5), 1896. doi:1896; https://doi.org/10.3390/su12051896

[9] Kaur, N. K. (2019). A Review of Various Maze Solving Algorithms Based on Graph Theory. *International Journal for Scientific Research & Development, 6*(12), 431-434.Obtenido de https://www.researchgate.net/publication/331481380_A_Review_of_Various_Maze_Solving_Algorithms_Based_on_Graph_Theory

[10] Llambías, V., & Osimani, P. (2022). Planificación de trayectorias óptimas, tesis de grado. Uruguay. Obtenido de https://hdl.handle.net/20.500.12008/32631

[11] Ma, Z., Wan, W., Song, L., Liu, C., Liu, H., & Wu, Y. (2022). An Approach of Path Optimization Algorithm for 3D Concrete Printing Based on Graph Theory. *Applied Sciences, 12*(22), 11315. doi:https://doi.org/10.3390/app122211315

[12] Niemczyk, R., & Zawiślak, S. (2018). Review of Maze Solving Algorithms for 2D Maze and Their Visualisation. *Springer*, 239-252.

[13] Tomás-Mariano, V. T., Núñez-Cárdenas, F. d., & Andrade-Hernández, E. (2014). Propuesta de construcción dinámica de espacios de búsqueda. *Res. Comput. Sci.*, 155-166.

[14] Trujillo-Romero, F. (2022). Planificación de trayectorias usando metaheurísticas. Visión electrónica, 16(1).

[15] Vargas, J. M., Pinzón, C. T., & Patiño, C. R. (2008). Técnicas de inteligencia artificial para la solución de laberintos de estructura desconocida. Scientia et technica, 2(39).