Actividad 1. Resolución de un problema mediante búsqueda heurística

José Miguel Chacón Ordóñez

Universidad Internacional de la Rioja, Logroño (España)

13 de Enero de 2022

RESUMEN

Durante el desarrollo de esta actividad se definirá, analizará y resolverá un problema de logística utilizando algoritmos de búsqueda informada y no informada y evaluando algunas variaciones, con el fin de comparar su comportamiento, diferencias y entender sus conceptos asociados con mayor profundidad.

PALABRAS CLAVE

búsqueda offline, búsqueda informada, definición de heurísticas

I. INTRODUCCIÓN

Encontrar caminos a un objetivo es una de las tareas más comunes que resolvemos en el día a día. En esta actividad haremos uso de algoritmos de búsqueda no informada: En amplitud y profundidad; y búsqueda heurística: A* para encontrar estos caminos, suponiendo un problema de una empresa de paquetería que describiremos con más detalle a continuación. El objetivo es caracterizar la situación adecuadamente, emplear los algoritmos mencionados, realizar algunas modificaciones al problema y a los parámetros de los algoritmos y, por último analizar y comparar los resultados de cada uno.

II. ANÁLISIS DEL PROBLEMA

El problema que nuestros agentes intentarán resolver consiste en lo siguiente: Una empresa de paquetería desea automatizar sus operaciones. Una de las tareas de la flota es entregar una serie de paquetes en distintos puntos de la ciudad. Para ello sus furgonetas automáticas tienen que ser capaces de encontrar caminos entre su ubicación y los puntos de recogida, conociendo el mapa de la ciudad.
Simplificamos el problema considerando el espacio dividido en una matriz rectangular, de modo que una furgoneta estará situada en una ubicación identificada por sus coordenadas. La furgoneta puede moverse en sentido horizontal y vertical, inicialmente asumiendo un coste real por casilla (g) de 1. El objetivo es llegar hasta la ubicación del paquete. El ejemplo del problema sobre el que vamos a trabajar es el indicado en la figura. II

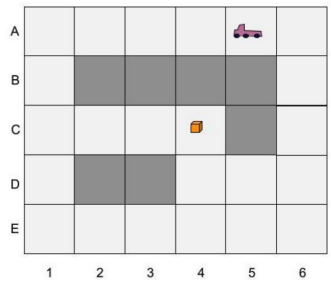

Figura 1: Mapa modelado del problema.

III. DESARROLLO. DE LA ACTIVIDAD

A. Búsqueda en amplitud

La búsqueda en amplitud o también llamada búsqueda primero en anchura es una estrategia en la que se expande primero el nodo raíz, a continuación todos sus sucesores, y después sus sucesores (etc). [1]. Es una búsqueda completa y óptima aunque representa una complejidad en tiempo y espacio muy elevada.

A.1. Asumiendo coste uniforme e igual a 1

A continuación veremos el paso a paso del desarrollo de la búsqueda en amplitud en un ejercicio de escritorio vs los resultados obtenidos usando simple-ai (python).
Para el caso de la ejecución del algoritmo manual, expandiremos cada nodo en orden alfabético y no repetiremos nodos visitados. En la figura 3 es posible ver el recorrido del agente (color naranja) en cada paso.

	Expandido	Se generan	Frontera
1	A5	A4,A6	A4,A6
2	A4	A3	A6,A3
3	A6	B6	A3,B6
4	A3	A2	B6,A2
5	B6	C6	A2,C6
6	A2	A1	C6,A1
7	C6	D6	A1,D6
8	A1	B1	D6,B1
9	D6	D5,E6	B1,D5,E6
10	B1	C1	D5,E6,C1
11	D5	D4,E5	E6,C1,D4,E5
12	E6		C1,D4,E5
13	C1	C2,D1	D4,E5,C2,D1
14	D4	C4,E4	E5,C2,D1,C4,E4
15	E5		C2,D1,C4,E4
16	C2	C3	D1,C4,E4,C3
17	D1	E1	C4,E4,C3,E1
18	C4	FIN	FIN

Figura 2: Ejecución paso a paso del problema usando BFS.

Para el caso de la ejecución usando python, podemos ver el camino encontrado en la figura 4. Podemos ver que para encontrar la meta realizó una iteración más (19), debido a que re-visitó un nodo. En el cuadro 1 están los resultados obtenidos.

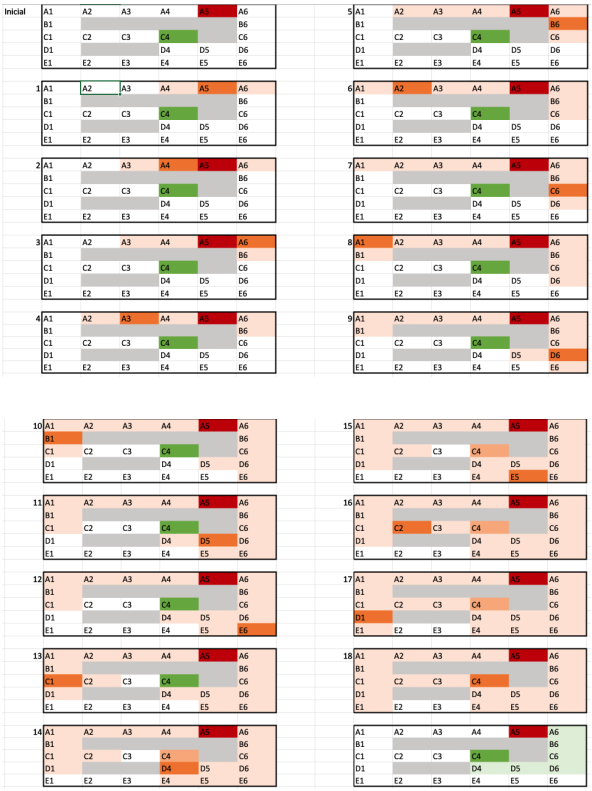

Figura 3: Acciones del agente en cada iteración.

Métrica	Resultado
Longitud	8
Coste	7.0
Frontera máxima	4
Nodos	19
Iteraciones	19

Cuadro 1: Resultados obtenidos para el coste uniforme en amplitud

A.2. Asumiendo coste no uniforme

En este caso se ejecutará la búsqueda asumiendo lo siguiente: los movimientos hacia abajo, izquierda y derecha tienen un coste de 1, mientras que los movimientos hacia arriba tienen un coste de 5. Al realizar la ejecución del algoritmo podemos notar que el coste asciende de 7 a 11, encontrando el mismo camino y por tanto, podemos concluir que tal aumento debe al movimiento que ejecuta el agente en la posición D4 hasta la meta.

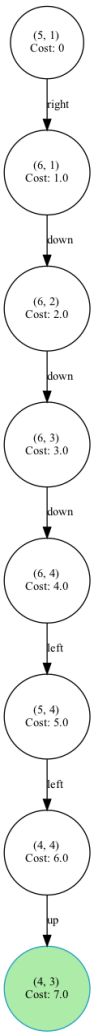

Figura 4: Camino encontrado usando BFS - Coste uniforme

Métrica	Resultado
Longitud	8
Coste	11.0
Frontera máxima	4
Nodos	19
Iteraciones	19

Cuadro 2: Resultados obtenidos para el coste no uniforme en amplitud

B. Búsqueda en profundidad

La búsqueda en profundidad siempre expande el nodo más profundo en la frontera actual del árbol de búsqueda. [1]

B.1. Asumiendo coste uniforme e igual a 1

Al igual que para la búsqueda en amplitud, realizaremos una ejecución de escritorio, expandiendo cada nodo en orden alfabético y sin repetir nodos visitados. En la figura B.1 es posible ver el recorrido del agente en cada paso.

	Expandido	Se generan	Frontera
1	A5	A4, A6	A4, A6
2	A4	A3	A3, A6
3	A3	A2	A2,A6
4	A2	A1	A1,A6
5	A1	B1	B1,A6
6	B1	C1	C1,A6
7	C1	C2,D1	C2,D1,A6
8	C2	C3	C3,D1,A6
9	C3	C4	C4,D1,A6
10	C4	FIN	FIN

Figura 5: Ejecución paso a paso del problema usando DFS.

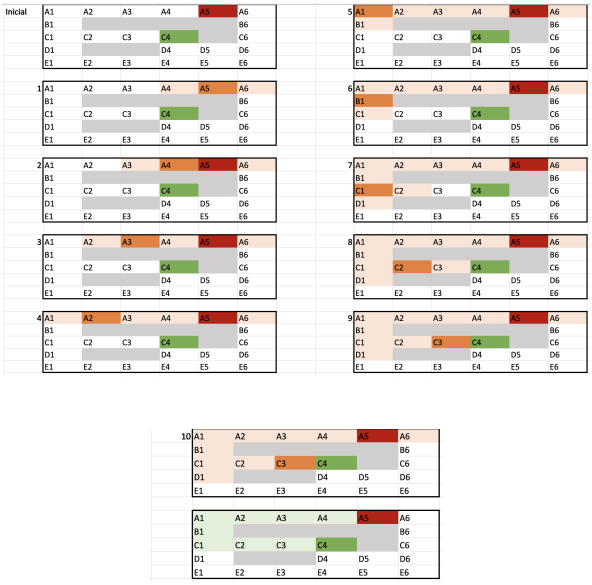

Figura 6: Acciones del agente en cada iteración.

Al ejecutar el algoritmo en código, la diferencia es más notable. Podemos ver que para encontrar la meta realizó más iteraciones (20), debido a que revisitó algunos nodos y el orden en que los eligió no fue el orden alfabético. En la figura B.1 se encuentra el grafo del camino hallado por el agente, y en el cuadro B.1 están los resultados obtenidos.

Métrica	Resultado
Longitud	8
Coste	7.0
Frontera máxima	6
Nodos	20
Iteraciones	20

Cuadro 3: Resultados obtenidos para el coste uniforme en profundidad

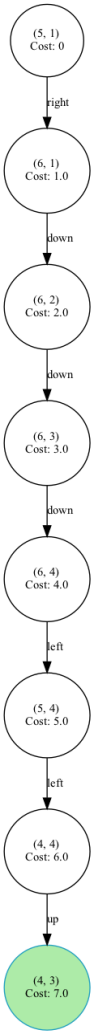

Figura 7: Camino encontrado en DFS

B.2. Expandiendo el menor número de nodos

En este caso evaluaremos el algoritmo de modo que tenga un estado inicial en el que obtengamos la solución óptima expandiendo menos nodos que el resto de los algoritmos acá expuestos, encontrando que el caso en que esto ocurre (empatando con A*) y siendo el valor mínimo de todos los posibles, en la posición inicial C1.

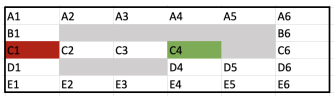

Figura 8: Posición inicial dónde DFS consigue el mejor resultado

Algoritmo	Nodos visitados
Amplitud	10
Profundidad	4
A* Manhattan	4
A* Euclídea	4

Cuadro 4: Comparación de nodos visitados con cada algoritmo

B.3. Asumiendo coste no uniforme

Al ejecutar la búsqueda asumiendo lo siguiente: los movimientos hacia abajo, izquierda y derecha con un coste de 1, mientras que los movimientos hacia arriba tienen un coste de 5, obtenemos un aumento de 5 en el coste en comparación con la ejecución anterior B.1, ya que el camino encontrado sólo ejecuta un movimiento hacia arriba B.3.

Métrica	Resultado
Longitud	8
Coste	11.0
Frontera máxima	6
Nodos	20
Iteraciones	20

Cuadro 5: Resultados obtenidos para el coste no uniforme en profundidad

C. Búsqueda A*

La búsqueda en A*, es la forma más ampliamente conocida de búsqueda primero el mejor. Es una búsqueda informada y hace uso de una función heurística para expandir cada nodo a su paso. En general, siempre intenta buscar el coste más barato de la solución tomando en

cuenta el nodo siguiente y un coste estimado para llegar al objetivo.

$$f(n) = g(n) + h(n)$$

Usando la distancia de Manhattan como función heurística.
En este caso nuestra función heurística está definida cómo:

$$h(n) = \sum_{i=1}^{n} |p_i - q_i|$$

C.1. Asumiendo coste uniforme e igual a 1

Asumiendo que el coste de cada movimiento es igual a 1, ejecutamos el algoritmo de búsqueda obteniendo el siguiente resultado.

Métrica	Resultado
Longitud	8
Coste	7.0
Frontera máxima	5
Nodos	11
Iteraciones	11

Cuadro 6: Resultados obtenidos para el coste uniforme en A* usando la distancia de Manhattan

C.2. Asumiendo coste no uniforme

En esta ejecución, el coste de mover el agente hacia arriba es igual a 5, mientras que el resto de movimientos tienen un valor igual de 1.

Métrica	Resultado
Longitud	10
Coste	9.0
Frontera máxima	5
Nodos	19
Iteraciones	19

Cuadro 7: Resultados obtenidos para el coste no uniforme en A* usando la distancia de Manhattan

Usando la distancia Euclídea como función heurística.

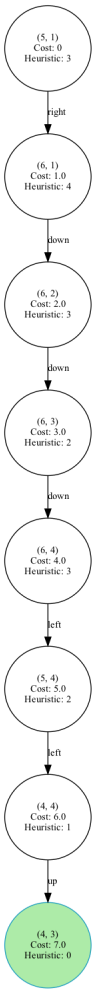

Figura 9: Camino encontrado usando A* con heurística de Manhattan - Coste uniforme

En este caso nuestra función heurística está definida cómo:

$$h(n) = \sqrt[2]{p_i^2 - q_i^2}$$

C.3. Asumiendo coste uniforme e igual a 1

Asumiendo que el coste de cada movimiento es igual a 1, ejecutamos el algoritmo de búsqueda obteniendo el siguiente resultado.

C.4. Asumiendo coste no uniforme

En esta ejecución, nuevamente el coste de mover el agente hacia arriba es igual a 5, mientras que el resto de movimientos tienen un valor igual de 1.

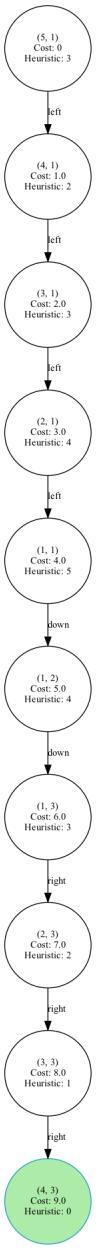

Figura 10: Camino encontrado usando A* con heurística de Manhattan - Coste no uniforme

Métrica	Resultado
Longitud	8
Coste	7.0
Frontera máxima	5
Nodos	11
Iteraciones	11

Cuadro 8: Resultados obtenidos para el coste uniforme en A* usando la distancia Euclídea

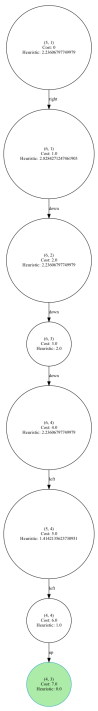

Figura 11: Camino encontrado usando A* con heurística Euclídea - Coste uniforme

Métrica	Resultado
Longitud	10
Coste	9.0
Frontera máxima	5
Nodos	19
Iteraciones	19

Cuadro 9: Resultados obtenidos para el coste no uniforme en A* usando la distancia Euclídea

IV. COMPARACIÓN DE RESULTADOS Y CONCLUSIONES

Se puede observar que el comportamiento de los algoritmos de búsqueda no informados (amplitud y profundidad) permanecieron iguales al modificar los costos, variando en sus resultados únicamente el coste final, debido a que no toman en cuenta esta variable al planear su ejecución. También encontramos que para este mapa las heurísticas de la distancia de Manhattan y la euclídea son válidas y se comportan similarmente en cuanto a nodos visitados e iteraciones.

Figura 12: Camino encontrado usando A* con heurística Euclídea - Coste no uniforme

Por último, durante el análisis y solución de el problema planteado se encontró que los criterios para resolver un problema deben estar claramente definidos desde el principio, ya que tienen un impacto fundamental en la resolución de este. Para estos algoritmos específicamente debemos definir con anterioridad las convenciones de expansión de los nodos (orden) y si el algoritmo revisitará nodos; y en caso de utilizar librerías o software de terceros debemos conocer o definir claramente sus reglas con el fin de evitar confusiones o ambiguedades.

V. DIFICULTADES ENCONTRADAS

En la búsqueda en profundidad se encontró una diferencia notable de los resultados al realizar la prueba de escritorio y el código python usando simple-ai. Después de examinar el log de la ejecución del código se encontró que ambas soluciones eran válidas.

A. APÉNDICES

Se adjunta el código python modificado y ejecutado para el desarrollo de la actividad.

Referencias

[1] Stuart J. Russell. *Inteligencia artificial, un enfoque moderno*, volume 3 1, page Capitulo 3. Ed. Prentice Halll, 2009.