Universidad
Carlos III de Madrid

Búsqueda Heurística en Radares

AUTORES:

Iván Ciller López: 100522245
100522245@alumnos.uc3m.es

Sergio Villafuertes Jiménez: 100522323
100522323@alumnos.uc3m.es

Jhonatan Barcos Gambaro: 100548615
100548615@alumnos.uc3m.es

GRUPO: 1

Inteligencia Artificial

Índice

1. Introducción

Esta memoria explicará con detalles cómo hemos realizado esta práctica, la cual está enfocada en proponer y desarrollar una solución a un problema de planificación de rutas para un avión espía cuyo objetivo es el de sobrevolar y fotografiar diferentes zonas dentro de un territorio vigilado por radares.

Buscamos desarrollar una búsqueda heurística que minimice la exposición de dicho avión frente a los sistemas radares de vigilancia. Para ello partimos de un mapa georreferenciado a modo de malla así como una configuración arbitraria de radares dentro de dicha malla.

Nuestro objetivo es el de llevar a cabo dicha solución a través de un sistema informático implementado en Python, a partir del código base proporcionado en el Aula Global, el cual ha sido modificado añadiendo todos los módulos necesarios para los siguientes puntos:

- Generación del mapa de detección.
- Construcción de la malla de navegación.
- Planificación de rutas.

Así mismo, se han diseñado dos heurísticas admisibles para este problema y se ha evaluado nuestro sistema en diferentes escenarios para verificar la correcta implementación en base a distintos parámetros variando el número de radares, resolución del mapa y umbral de tolerancia.

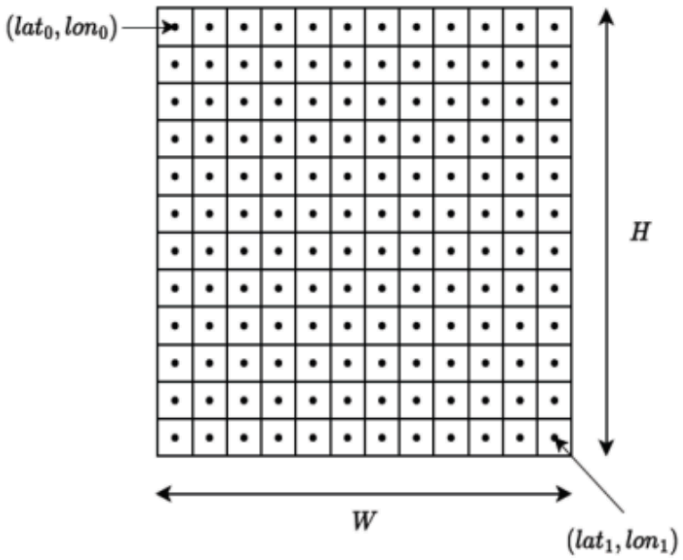
A través de esta memoria buscamos una detallada descripción de la solución propuesta cumplimentando todos los puntos del enunciado.

2. Contenido

2.1. Modelado del problema de búsqueda

2.1.1. Estados y operadores del problema

En primer lugar definamos los estados y operadores del problema introducido. Para proponer una solución hemos trabajado con un modelado sobre una malla bidimensional de tamaño *H* x *W*, siendo *H* la altura de la malla y *W* la anchura de la misma. Cada celda de la malla representa un posible estado dentro de nuestro sistema. Así mismo, podemos definir la representación de un estado *s* como:



$$s = (i, j)$$
$$i \in [0, H)$$
$$j \in [0, W)$$

Por otra parte, a cada celda de la malla se le asigna una probabilidad de detección:

$$\Psi(x_1, x_2, \ldots, x_n) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

$$\Psi^*(lat, lon) = máx_{\Psi_i^*}\{\Psi_i^*(lat, lon)\}_{i=1}^{N_r}$$

$$\Psi_i^*(lat, lon) = \{\Psi_i(lat, lon)\ si\ d \leq R_{máx}\ 0\ si\ d > R_{máx}$$

$$\Psi^*_{scaled} = \frac{\Psi^* - \Psi^*_{mín}}{\Psi^*_{máx} - \Psi^*_{mín}}(1 - \epsilon) + \epsilon$$

La cual corresponde al riesgo de ser detectado por al menos uno de los radares del entorno del avión. Dicha probabilidad es un número entre ε y 1, calculada como el máximo de las detecciones individuales de cada uno de los posibles radares presentes en la malla, aplicando una normalización MinMax a posteriori.

Respecto a los operadores de nuestro problema, definidos como los desplazamientos en cada una de las posibles direcciones de la malla son los siguientes:

- Arriba: (i - 1, j)
- Abajo: (i + 1, j)

- Izquierda: (i, j - 1)
- Derecha: (i, j + 1)

Nótese que un operador solo lo consideramos aplicable en caso de que la celda de destino tenga una probabilidad de detección (definida previamente) menor o igual a la tolerancia del sistema.

2.1.2. Estado inicial y final

Por una parte, el estado inicial ha sido definido como el primer punto de interés, a partir de ahora POI, transformando a coordenadas discretas mediante el método *discretize_coords*, el cual convierte el par (latitud, longitud) a los índices (i,j) de la malla de nuestro modelo.

Por otra parte, el estado/s final/es no son definidos como únicos sino que los tratamos como una secuencia de estados intermedios. Así, podemos definir la ruta completa como la concatenación de todos los caminos calculados entre cada par de POIs en orden. Un ejemplo matemático de la planificación de una ruta completa para nuestro problema sería la siguiente:
$A * (p(1), p(2)) + A * (p(2), p(3)) + \ldots + A * (p(n-1), p(n))$

2.1.3. Heurísticas propuestas

Nuestra propuesta se basa en dos funciones heurísticas admisibles, las cuales justificamos a continuación.

2.1.3.1. Heurística h1: Basada en la Distancia de Manhattan

$$d_M(x,y) = \sum_{i=1}^{d} |x_i - y_i| \Rightarrow d = 2 \Rightarrow |x_1 - y_1| + |x_2 - y_2|$$

Para ajustar dicha heurística a nuestro problema, realizamos el producto de dicho valor dM(x,y) por el coste mínimo ε, tal que nuestra heurística h1 = dM(x,y) * ε.

A continuación, justificamos la admisibilidad de dicha heurística:
Dicha heurística mide el número mínimo de movimientos necesarios en nuestro modelo de rejilla 4-conexa desplazándose con los operadores disponibles definidos previamente. Así mismo, definimos el coste mínimo por paso como ε , representando así h1 el coste más bajo posible en el caso de que cada movimiento tuviera un coste exacto ε. Podemos concluir entonces que h1(n) <= h*(n) y por tanto, h1 es admisible ya que el coste real nunca podrá ser menor que ε por celda.

2.1.3.2. Heurística h2: Basada en la Distancia Euclídea

$$d_E(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} \Rightarrow d = 2 \Rightarrow \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Para ajustar dicha heurística a nuestro problema, realizamos el producto de dicho valor dE(x,y) por el coste mínimo ε, tal que nuestra heurística h1 = dE(x,y) * ε.

A continuación, justificamos la admisibilidad de dicha heurística:
Dicha heurística mide la distancia en línea recta entre un par de celdas. A pesar de que no existe ningún estado que capacite a nuestro sistema el moverse en diagonal dentro de la malla, esta distancia siempre será menor o igual que la longitud del camino más corto utilizando los operadores disponibles (horizontales y/o verticales). Así mismo, definimos el coste mínimo por paso como ε, representando así h2 el coste más bajo posible en el caso de que cada movimiento tuviera un coste exacto ε. Podemos concluir entonces que $h2(n) \leq h^*(n)$ y por tanto, h2 es admisible ya que el coste real nunca podrá ser menor que ε por celda aunque no siempre representará una trayectoria viable.
Nótese que el objetivo de dichas heurísticas es el de utilizarlas junto al algoritmo A*, el cual recordamos que dada una heurística admisible, asegura encontrar una solución completa y óptima para nuestra práctica.

2.2. Generación del mapa y espacio de búsqueda

2.2.1. Generación del mapa

Para llevar a cabo la generación del mapa de nuestro modelo utilizamos la función *compute_detection_map* implementada en *Map.py,* la cual recorre cada celda del mapa y calcula la probabilidad de detección de cada celda de la siguiente forma:

En primer lugar, para cada uno de los radares del modelo, realiza una evaluación de su nivel de detección a través de una gaussiana bidimensional centrada en su ubicación, aplicable en el caso de que la distancia sea menor que el radio máximo de detección Rmax. Seguidamente, para cada (i, j) se selecciona la detección máxima entre todos los radares y se aplica una normalización para evitar valores anómalos.

Finalmente, una vez generado el mapa, lo utilizaremos como base para la construcción gráfica del espacio de búsqueda, la cual se detalla a continuación.

Nótese que la descripción matemática de la implementación de dicha función se ha explicitado en el punto [2.1.1] de esta memoria.

2.2.2. Implementación de la generación gráfica

Una vez realizada la generación del mapa, se construye el grafo a través de la función *build_graph* implementada en *SearchEngine.py*, en la cual utilizamos la librería *networkx.DiGraph* para crear un grafo dirigido teniendo en cuenta las siguientes pautas:

- Cada uno de los nodos representan una celda accesible, es decir, con valores dentro de la tolerancia definida en el sistema.
- Se generan artistas entre cada uno de los vecinos directos según los operadores definidos previamente.
- El peso de cada arista corresponde al valor de detección de la celda destino.

Así pues, una vez creado el grafo en función de estas pautas, ya podemos aplicar algoritmos de búsqueda A* para encontrar las rutas más óptimas para nuestro problema entre POIs.

2.3. Integración del algoritmo de búsqueda A*

Una vez realizada la generación gráfica del mapa de nuestro modelo, ya tenemos todo lo necesario para utilizar el algoritmo de búsqueda A*, el cual ha sido definido en *Main.py*. Se detalla su funcionamiento a continuación:

En primer lugar, se realiza la lectura de los parámetros impuestos sobre nuestro modelo desde *scenarios.json.* Seguidamente, se lleva a cabo la construcción de los límites *Boundaries* y del mapa *Map,* así como la generación de forma aleatoria de los radares y el mapa de detección. Así mismo, se lleva a cabo la construcción gráfica con la función *build_graph,* se discretizan los POIs con la función *discretize_coords* y se verifica la accesibilidad de los POIs en el grafo creado. Finalmente, se aplica el algoritmo de A* entre cada par de POIs utilizando una de las heurísticas propuestas y se almacena en forma de lista la solución devuelta por el algoritmo como la secuencia de nodos sinónimo de ruta óptima. Se calcula el coste total de dicha ruta a través de la función *compute_path_cost* y se visualizan los resultados con librerías de *matplotlib.*

3. Experimentación

En esta sección se encuentran los experimentos realizados para cada uno de los distintos escenarios planteados en el enunciado. Para cada uno de los escenarios hemos planteado 3 casos diferentes, utilizando diferentes tolerancias para demostrar cuándo se encuentran las rutas y, en caso de hacerlo, mostrando el coste total y el número de nodos expandidos para ello.

3.1. Scenario_0

Escenario	Tamaño	Radares	POIs	Tolerancia	Coste total	Nodos expandidos	Ruta	Comentarios
scenario_0	16x16	1	2	0.2	-	-	No	El POI (np.int32(0), np.int32(9)) no está accesible (fuera del grafo por alta detección).
scenario_0	16x16	1	2	0.5	0.5789	243	Sí	Ruta directa, coste bajo.
scenario_0	16x16	1	2	0.9	0.5789	249	Sí	Misma ruta que con 0.5, coste constante.

3.2. Scenario_1

Escenario	Tamaño	Radares	POIs	Tolerancia	Coste total	Nodos expandidos	Ruta	Comentarios
scenario_1	16x16	3	2	0.2	-	-	No	El POI (np.int32(0), np.int32(0)) no está accesible (fuera del grafo por alta detección).
scenario_1	16x16	3	2	0.5	1.2043	226	Sí	Ruta más evasiva, poco costosa.
scenario_1	16x16	3	2	0.9	1.2043	229	Sí	Mismo coste, ligera variación en nodos.

3.3. Scenario_2

Escenario	Tamaño	Radares	POIs	Tolerancia	Coste total	Nodos expandidos	Ruta	Comentarios
scenario_2	32x32	7	4	0.2	-	-	No	El POI (np.int32(30), np.int32(15)) no está accesible (fuera del grafo por alta detección).
scenario_2	32x32	7	4	0.5	4.7204	1205	Sí	Ruta larga, bordeando múltiples radares.
scenario_2	32x32	7	4	0.9	4.7204	1252	Sí	Mismo camino, ligero aumento en expansión.

3.4. Scenario_3

Escenario	Tamaño	Radares	POIs	Tolerancia	Coste total	Nodos expandidos	Ruta	Comentarios
scenario_3	32x32	10	4	0.2	-	-	No	El POI (np.int32(30), np.int32(15)) no está accesible (fuera del grafo por alta detección).
scenario_3	32x32	10	4	0.5	8.4723	1243	Sí	Ruta más larga y costosa.
scenario_3	32x32	10	4	0.9	8.4441	1370	Sí	Mismo camino, mayor expansión.

3.5. Scenario_4

Escenario	Tamaño	Radares	POIs	Tolerancia	Coste total	Nodos expandidos	Ruta	Comentarios
scenario_4	32x32	16	4	0.2	-	-	No	El POI (np.int32(12), np.int32(20)) no está accesible (fuera del grafo por alta detección).
scenario_4	32x32	16	4	0.5	10.6885	1197	Sí	Ruta larga y costosa.
scenario_4	32x32	16	4	0.9	10.4615	1340	Sí	Coste ligeramente menor, más nodos expandidos.

3.6. Scenario_5

Escenario	Tamaño	Radares	POIs	Tolerancia	Coste total	Nodos expandidos	Ruta	Comentarios
scenario_5	64x64	16	6	0.2	-	-	No	El POI (np.int32(0), np.int32(59)) no está accesible (fuera del grafo por alta detección).
scenario_5	64x64	16	6	0.5	59.1621	6496	Sí	Ruta larga, con más POIs y mayor mapa.
scenario_5	64x64	16	6	0.9	48.4735	6434	Sí	Coste menor al aceptar más riesgo.

3.7. Scenario_6

Escenario	Tamaño	Radares	POIs	Tolerancia	Coste total	Nodos expandidos	Ruta	Comentarios
scenario_6	64x64	20	6	0.2	-	-	No	El POI (np.int32(59), np.int32(30)) no está accesible (fuera del grafo por alta detección).
scenario_6	64x64	20	6	0.5	27.1325	5909	Sí	Ruta con coste moderado y muchas expansiones.
scenario_6	64x64	20	6	0.9	27.1325	5965	Sí	Mismo coste, más nodos explorados.

3.8. Scenario_7

Escenario	Tamaño	Radares	POIs	Tolerancia	Coste total	Nodos expandidos	Ruta	Comentarios
scenario_7	128x128	20	6	0.2	-	-	No	El POI (np.int32(0), np.int32(66)) no está accesible (fuera del grafo por alta detección).
scenario_7	128x128	20	6	0.5	37.8398	24832	Sí	Escenario más grande, muchas expansiones.
scenario_7	128x128	20	6	0.9	37.8398	25475	Sí	Mismo coste, más nodos expandidos.

3.9. Scenario_8

Escenario	Tamaño	Radares	POIs	Tolerancia	Coste total	Nodos expandidos	Ruta	Comentarios
scenario_8	256x256	32	6	0.2	-	-	No	El POI (np.int32(0), np.int32(242)) no está accesible (fuera del grafo por alta detección).
scenario_8	256x256	32	6	0.5	-	-	No	El POI (np.int32(109), np.int32(242)) no está accesible (fuera del grafo por alta detección).
scenario_8	256x256	32	6	0.9	131.3659	89219	Sí	Ruta muy costosa y con altísima expansión.

3.10. Scenario_9

Escenario	Tamaño	Radares	POIs	Tolerancia	Coste total	Nodos expandidos	Ruta	Comentarios
scenario_9	1024x1024	20	20	0.2	-	-	No	El POI (np.int32(457), np.int32(118)) no está accesible (fuera del grafo por alta detección).
scenario_9	1024x1024	20	20	0.5	-	-	No	El POI (np.int32(457), np.int32(118)) no está accesible (fuera del grafo por alta detección).
scenario_9	1024x1024	20	20	0.9	3968.2670	11753336	Sí	Ruta con coste y expansión masivos.

4. Conclusiones

Tras haber explicado cómo hemos diseñado todo el código correspondiente, hemos podido entender adecuadamente la práctica y los resultados obtenidos. Tanto el mapa de detección, como la malla de navegación, han sido elaborados correctamente. Esto ha sido posible gracias a los métodos empleados en las funciones del código necesitadas de ser completadas, además de las heurísticas que hemos diseñado que han facilitado todo ello. Además, hemos seguido una adecuada planificación de rutas, que nos ha permitido llevar a cabo correctamente los experimentos de los escenarios planteados como ejemplo del código base.

Respecto a estos casos de prueba, hemos podido deducir que a medida que aumentan la densidad de radares y el tamaño del mapa, la dificultad para encontrar rutas seguras crece significativamente. Por otro lado, la tolerancia es un parámetro que indica que, cuanto menor sea, más complicado será encontrar rutas viables, como hemos podido observar principalmente en los casos con tolerancia 0.2 y 0.5 en algunos casos debido a lo comentado anteriormente. Por el contrario, cuanto mayor es la tolerancia establecida, mayor es la facilidad para encontrar rutas, con mayor coste total y número de nodos expandidos en función del tamaño del escenario.

Finalmente, podemos concluir que se han gestionado correctamente los casos en los que un POI cae en una zona bloqueada, evitando errores y mostrando la información de interés al usuario; y los resultados obtenidos permiten validar el comportamiento del sistema ante distintos niveles de riesgo, escalabilidad y configuración de radares, cumpliendo los objetivos de la práctica.

5. Declaración de uso de Inteligencia Artificial generativa

5. 1. Para qué aspectos de la práctica se ha empleado la IA

Para llevar a cabo la práctica hemos utilizado herramientas de Inteligencia Artificial generativa como ChatGPT de OpenAI en combinación de técnicas de Prompt Engineering para maximizar la utilidad y correctitud de estas herramientas. A continuación desglosamos los principales puntos para los que hemos utilizado este tipo de IA:

- Comprensión del enunciado de la práctica: En primer lugar, hemos utilizado IA generativa para garantizar una correcta y total comprensión del enunciado de la práctica a nivel técnico y matemático, especialmente para las fórmulas de detección y radares. Así mismo, hemos podido asentar el enunciado a través de ejemplos prácticos para poder desarrollar la práctica con total garantías a nivel de entendimiento.

- Resolución de dudas frente a los problemas surgidos durante la elaboración de la práctica: A medida que íbamos resolviendo el enunciado planteado, surgían distintos problemas a nivel de código como errores de ejecución, los cuales solucionamos fácilmente a través del debugging de VSCode junto con la ayuda de IA generativa para detectar en qué parte de nuestro código se encontraba el error para así poder solucionarlo.

- Revisión del código y validación de nuestro modelo: Una vez finalizado nuestro código, hemos realizado un análisis exhaustivo del mismo para comprobar que éste era correcto. De hecho, gracias a esta tecnología nos hemos dado cuenta de que teníamos un error matemático a la hora de definir las funciones de la heurística 2 en *SearchEngine.py* y en *compute_max_range()* en *Radar..py,* en las que en vez de elevar al cuadrado (**2) estabamos realizando el producto (*2).

5. 2. En qué medida le ha facilitado la realización de la práctica

En general, el acceso a estas nuevas herramientas de forma responsable y coherente han supuesto una gran ventaja a la hora de realizar la práctica, sobretodo en términos de tiempo, ya que nos ha facilitado diferentes aspectos como podrían ser:

- Rápida comprensión de conceptos complejos, para los cuales, en un pasado tardaríamos mucho más tiempo en encontrar la información indicada en internet, interpretarla y adaptarla a nuestras necesidades.

- Resolución de errores, para los cuales, al igual que el punto anterior, en un pasado podríamos tardar mucho en detectar y solventar de manera correcta.

- Obtener explicaciones sobre los métodos de librerías de python que no conocemos de una forma mucho más rápida y eficaz que consultando la guía de las mismas.

- Verificación de contenido de la memoria para comprobar que no nos dejamos nada relevante por detallar para el total entendimiento de nuestro trabajo desenvuelto en dicha práctica.

5. 3. Cómo ha verificado y adaptado la información obtenida

Como se ha comentado en el anterior apartado, consideramos que el uso de estas nuevas herramientas son una ventaja muy grande respecto al pasado, siempre y cuando se utilicen de manera responsable y concienzuda. Por ello, hemos utilizado un método de validación cruzada para asegurarnos de que la información ofrecida por la IA generativa era correcta y óptima. Para ello, se ha seguido la siguiente metodología:

- Comprobación sobre la validez de las respuestas a través de Internet y recursos proporcionados sobre Python, Numpy y enunciado de la práctica.

- Contrastación de las explicaciones teóricas con el enunciado y otras fuentes, especialmente en la parte matemática ya que las IAs generativas no están perfeccionadas respecto a este ámbito.

- Utilización de la IA como herramienta de apoyo y no como sustituto frente al pensamiento crítico para resolver los distintos problemas planteados.