

Inteligencia artificial



Control de un dron de reconocimiento

Grupo 82

Laura Álvarez Flórez
Laura Yunta García

100363965
100363785

100363965@alumnos.uc3m.es
100363785@alumnos.uc3m.es

Contenido

- 1. Introducción 3
- 2. Problema básico 3
 - a. Diseño realizado 3
 - b. Experimentación básica 4
 - c. Experimentación con nuevos escenarios 5
- 3. Parte Avanzada..... 7
 - a. Diseño realizado 7
 - b. Experimentación en cada uno de los escenarios y análisis..... 8
- 5. Conclusiones y comentarios..... 12

1. Introducción

Este archivo documenta el proyecto final de Inteligencia Artificial: "Control de un dron de reconocimiento".

Se ha dividido el proyecto en dos partes, la primera parte se trata de un problema simplificado donde el dron ejecuta su cometido de manera simple, en la segunda parte se le han añadido funcionalidades más avanzadas, ambas están descritas con detalle en los apartados 2 y 3 del documento respectivamente.

2. Problema básico

a. Diseño realizado

Para implementar el problema básico hemos establecido el siguiente diseño:

Estado: (<posición del Dron>, <posición fotos>)

Estado Final: (<posición del Dron Inicial>, ((-1,-1), (-1,-1)...))

Tanto la posición del Dron como la de las fotos son tuplas con sus correspondientes coordenadas.

Algoritmia: El problema base sigue un algoritmo muy simple, se establece la posición del Dron en cada momento y además la posición de las fotos, cuando ejecutamos el algoritmo de búsqueda y el Dron alcanza una foto dicha tupla se modifica poniendo una posición invalida.

Sabemos que el estado final se ha completado porque todas las posiciones de las fotos tienen dicho valor por defecto y el Dron ha retornado a la posición inicial.

El movimiento correcto del Dron se lleva a cabo a través de las acciones de este. Durante el proceso se comprueba si el siguiente estado que se quiere alcanzar, es decir, la siguiente posición en el grid es accesible. Hemos determinado como accesibles cualquier casilla que no sea mar, controlando también los límites del grid establecidos.

Si una acción no es accesible en base a esto se elimina de la lista.

Heurística: Para la heurística del problema hemos probado diferentes implementaciones (las cuales serán analizadas más adelante en el documento) y finalmente hemos optado por establecer para el análisis la que mejor resultado nos proporcionaba a la que hemos denominado: Manhattan Ave.

Manhattan Ave es una adaptación de la Heurística "La distancia de Manhattan" para este problema. Determinamos el número de pasos mínimo que el Dron debe dar para alcanzar cada foto restante y a continuación estimamos el mínimo de estos datos.

b. Experimentación básica

Para cerciorarnos del correcto funcionamiento tanto del problema como de la heurística propuesta hemos realizado una batería exhaustiva de pruebas.

Hemos comenzado el experimento analizando el mapa proporcionado en el ejemplo, se le han aplicado todos los algoritmos de búsqueda proporcionados por la librería SIMPLE-AI para cerciorarnos de que la implementación es correcta. Además, hemos aplicado dichos algoritmos con y sin la heurística diseñada, quedándose demostrado que la heurística es admisible y proporciona resultados positivos.

En la siguiente tabla se puede observar una comparativa completa de dichas pruebas:

Algoritmo de búsqueda	A-STAR *	BREADTH FIRST	GREEDY *	ITERATIVE LIMITED DEPTH FIRST	UNIFORM COST	DEPTH FIRST
Sin Heurística	total length of solution: 12 total cost of solution: 12 visited nodes: 489 iterations: 489 max fringe size: 85	total length of solution: 12 total cost of solution: 12 visited nodes: 500 iterations: 500 max fringe size: 83	total length of solution: 24 total cost of solution: 24 visited nodes: 252 iterations: 252 max fringe size: 68	total length of solution: 14 total cost of solution: 14 visited nodes: 2906 iterations: 2906 max fringe size: 27	total length of solution: 12 total cost of solution: 12 visited nodes: 489 iterations: 489 max fringe size: 85	total length of solution: 38 total cost of solution: 38 visited nodes: 91 iterations: 91 max fringe size: 53
Heurística Manhattan	total length of solution: 12 total cost of solution: 12 visited nodes: 518 iterations: 518 max fringe size: 94	total length of solution: 12 total cost of solution: 12 visited nodes: 500 iterations: 500 max fringe size: 83	total length of solution: 12 total cost of solution: 12 visited nodes: 465 iterations: 465 max fringe size: 101	total length of solution: 14 total cost of solution: 14 visited nodes: 2906 iterations: 2906 max fringe size: 27	total length of solution: 12 total cost of solution: 12 visited nodes: 489 iterations: 489 max fringe size: 85	total length of solution: 38 total cost of solution: 38 visited nodes: 91 iterations: 91 max fringe size: 53

Como se puede observar en la tabla al ejecutar el juego, las búsquedas que utilizan una heurística en su algoritmo (señaladas con un asterisco [*]) han ejecutado el juego llegando a la solución con una longitud menor en el caso en el que se ha implementado la heurística diseñada en comparación con la ausencia de una heurística.

Por otro lado, si comparamos los algoritmos entre si podemos observar que, en la ausencia de una heurística que guíe la búsqueda, el coste del problema y el número de nodos expandidos varía considerablemente.

La búsqueda A-STAR, BREADTH FIRST y UNIFORM COST nos proporcionan unos mejores resultados siendo el coste 12 casillas y por detrás hablando en términos de coste se situarían ITERATIVE LIMITED DEPTH FIRST, GREEDY y por ultimo DEPTH FIRST cuyo resultado se aleja considerablemente de los primeros algoritmos de búsqueda.

El número de nodos visitados por los algoritmos que proporcionaban un mayor coste también es una cota mayor, estos visitan una media de 493 nodos, además, los algoritmos que daban un peor resultado en la comparativa anterior: GREEDY y DEPTH FIRST presentan un menor número de nodos visitados como era previsible. Por otro lado, el algoritmo ITERATIVE LIMITED DEPTH FIRST realiza una visita de nodos muy intensa realizando 2906 accesos a nodos.

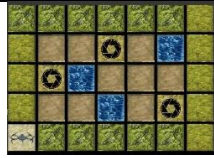
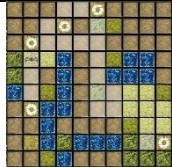
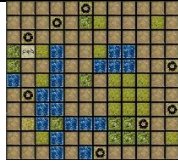
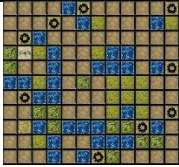
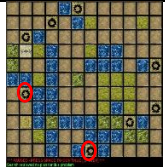
Finalizamos la primera comparativa del análisis del proyecto observando la memoria que ha sido utilizada por dichos algoritmos, los valores vuelven a ser similares para los tres algoritmos destacados (A-STAR, BREADTH FIRST y UNIFORM COST) haciendo un uso de esta elevado, en cambio, en este caso el algoritmo que mejor uso de la memoria realiza es ITERATIVE LIMITED DEPTH FIRST seguido de DEPTH FIRST y luego GREEDY.

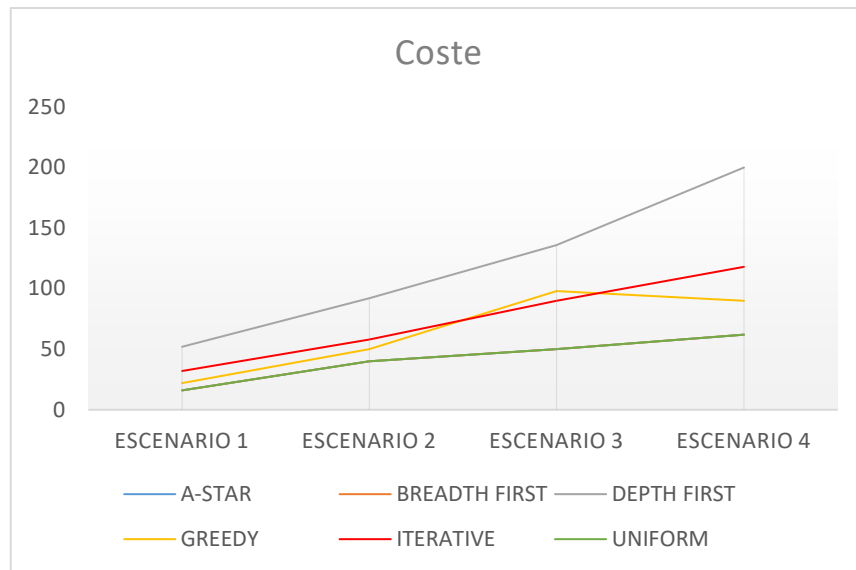
Como conclusión de este primer acercamiento al experimento observamos que los algoritmos que mejor coste proporcionan al problema del Dron también son los que mayor número de accesos a nodos realizan y por consiguiente mayor cantidad de memoria utilizada para resolver el problema de reconocimiento. Estas conclusiones son pares tanto si se usa o no una heurística.

c. Experimentación con nuevos escenarios

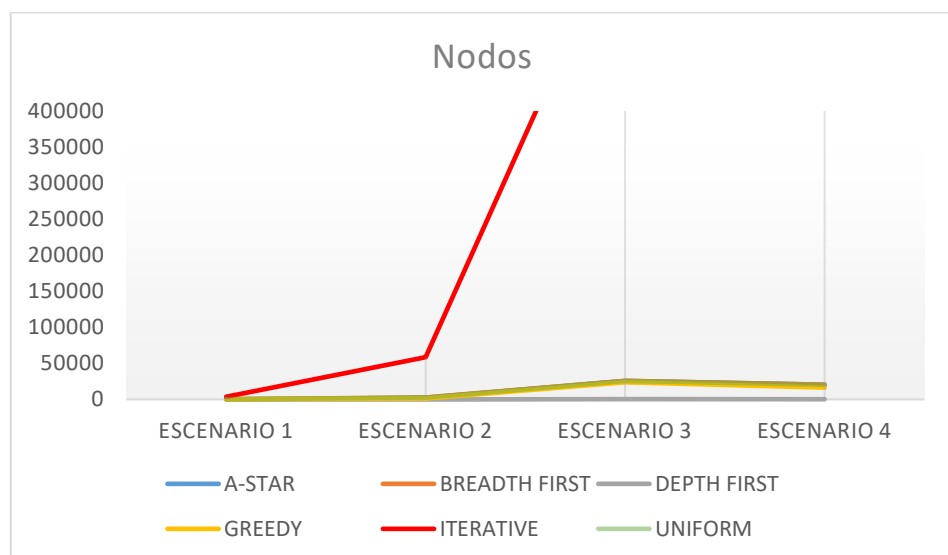
Para comprender el funcionamiento de nuestro Dron y comprobar que su control de reconocimiento muestra resultados positivos en cualquiera de los casos que se le puedan plantear, hemos diseñado 4 posibles escenarios.

Dado que no es el fin realizar una amplia gama de comprobaciones sino estimar la evolución de los algoritmos cuando el problema se vuelve más complejo, hemos decidido diseñar los escenarios prestando atención a características concretas que dificultan el problema como: número de fotos a realizar, tamaño del terreno y terrenos no accesibles (agua).

ESCENARIO 1	ESCENARIO 2	ESCENARIO 3	ESCENARIO 4	ESCENARIO X
				
Comprende el diseño más básico con un grid de 5x7 casillas, 3 fotos y 3 casillas tipo agua.	Se ha ampliado el escenario 1 a 10x10 casillas, 5 fotos y 17 casillas tipo agua.	Diseño más avanzado del grid con 11x12 casillas, 8 fotos y 20 casillas tipo agua.	Diseño similar al anterior con un grid de 11x12 casillas, 9 fotos y 29 casillas tipo agua. En este el aumento de dificultad reside en la colocación estratégica del agua.	Este escenario (no se ha analizado en los diagramas) es único ejemplo de un estado inicial que no se ha contemplado. No se puede acceder a las fotos por lo que no produce resultados.

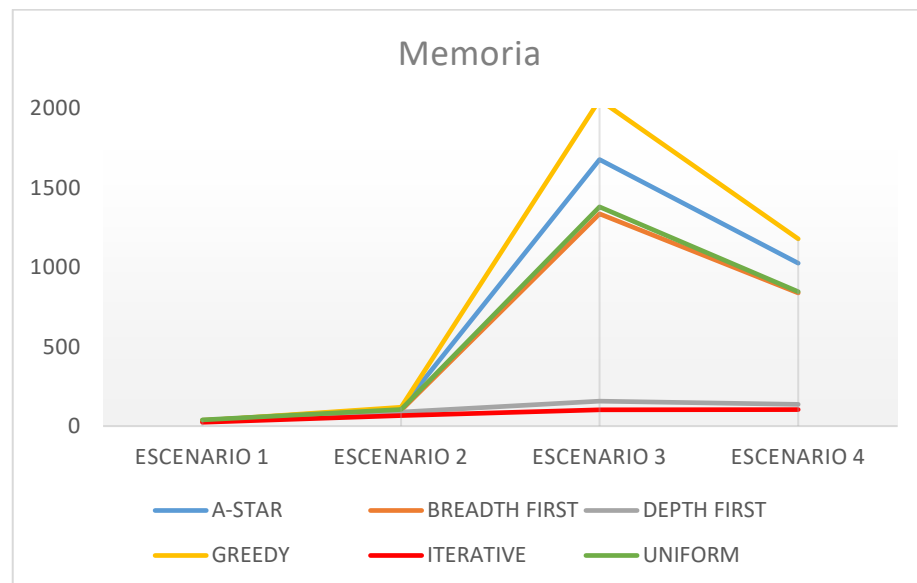


Análisis: Respecto a este gráfico comparativo de los valores coste de cada escenario respecto de cada algoritmo de búsqueda observamos que todos ellos crecen linealmente entre cada uno de los escenarios con más o menos rapidez dependiendo del algoritmo excepto el algoritmo GREEDY cuyo coste disminuye en el transcurso del ESCENARIO 3 al ESCENARIO 4. Por último cabe destacar, que en general el algoritmo más costoso en ejecución es el DEPTH FIRST y el menos costoso el UNIFORM COST.



Análisis: Respecto a este gráfico comparativo de los valores nodos visitados de cada escenario respecto de cada algoritmo de búsqueda podemos observar que todos los algoritmos tienen un comportamiento similar y con los mismos valores excepto el algoritmo el ITERATIVE LIMITED DEPTH FIRST que tendrá un

comportamiento independiente siempre ascendente. Finalmente cabe destacar que el peor algoritmo respecto a nodos es el ITERATIVE LIMITED DEPTH FIRST.



Análisis: Respecto a este gráfico comparativo de los valores coste de memoria de cada escenario respecto de cada algoritmo de búsqueda observamos que todos ellos crecen linealmente con bastante rapidez en el paso del ESCENARIO 2 al ESCENARIO 3 y disminuyen desde el ESCENARIO 3 al ESCENARIO 4. También se puede observar que los algoritmos ITERATIVE LIMITED DEPTH FIRST y DEPTH FIRST tienen un comportamiento totalmente distinto a los algoritmos anteriores de tal forma que su progresión prácticamente cumple el modelo de una constante. Por último cabe destacar que el algoritmo que más costes de memoria requiere es GREEDY y el que menos ITERATIVE LIMITED DEPTH FIRST.

3. Parte Avanzada

a. Diseño realizado

Algoritmia: El problema avanzado mantiene las funcionalidades del problema base, pero además se le han incluido nuevas funcionalidades. El Dron además de no poder acceder a las casillas “mar” tampoco puede continuar su camino si la batería esta en 0. Se ha añadido un nuevo tipo de casilla que representa una gasolinera (lugar donde el Dron recarga su batería al máximo inicial establecido) esta casilla permite al Dron recuperar su batería hasta el nivel máximo de nuevo. La batería del Dron se ha considerado como un gasto por unidad de tiempo, cuando avanza una casilla independientemente del tipo se considera que ha transcurrido una unidad por lo tanto se ve reflejado reduciéndose su batería dicha cantidad.

Además, en esta nueva versión se tiene en cuenta el coste, y en las casillas tipo “hill” se ha incrementado a 2 unidades.

Para desarrollar el problema avanzado hemos partido del diseño problema base y han sido realizadas las siguientes modificaciones para lograr un funcionamiento más complejo.

Estado: (<posición del Dron>, <posición fotos>, <batería>)

Estado Final: (<posición del Dron Inicial>, ((-1,-1), (-1,-1)...), (-1, <batería Max>)

Ahora el Estado cuenta con una tupla mas, esta es la encargada de establecer el nivel de batería del Dron actual y el nivel máximo de carga que puede llegar a alcanzar.

El movimiento correcto del Dron se lleva a cabo a través de las acciones de este. Durante el proceso se comprueba si el siguiente estado que se quiere alcanzar, es decir, la siguiente posición en el grid es accesible y además se analiza el nivel de batería, teniendo que ser este mayor que 0 si se quiere continuar. Si una acción no es accesible en base a esto se elimina de la lista.

Heurística: El análisis de las heurísticas está reflejado en el siguiente punto del documento, hemos optado por establecer como determinada: Manhattan 5 Ave. Manhattan 5 Ave es una versión mejorada de la anterior. Continuamos determinando el número de pasos mínimo que el Dron debe dar para alcanzar cada foto restante y a continuación estimamos el mínimo de estos datos, pero además si todas las fotos han sido tomadas se realiza el mismo algoritmo para que el Dron retorne a casa. Calculándose dicha distancia “relajada” sin tener en cuenta las casillas no accesibles.

b. Experimentación en cada uno de los escenarios y análisis

Para seguir analizando y comprobando las últimas funcionalidades implementadas para el nuestro Dron y comprobar que su control de reconocimiento muestra resultados positivos en cualquiera de los casos que se le puedan plantear, hemos diseñado 2 posibles escenarios de diversas complejidades, de tal forma que gracias a estos se puede comprobar y estimar la evolución de los algoritmos en función de los incrementos de complejidad.

Los parámetros variables de cada problema serán: el número de fotos a realizar, el número de puntos de recarga de la pila y los terrenos no accesibles (agua).

Inicialmente hemos decidido realizar un análisis de las 4 heurísticas propuestas en nuestro diseño:

HEURISTICAS	# FOTOS	#FOTOS (MEJORADA)	MANHATTAN AVE	MANHATTAN 5 AVE
Descripción	Numero de fotos restantes por realizar hasta alcanzar el estado final	Igual que # FOTOS pero añadiendo una unidad mas. Esta representa la vuelta a la base	Minimo del calculo de las distancias de Manhattan	Igual que Manhattan ave pero considerando la vuelta del agent a la base
A-STAR	total length of solution: 36 total cost of solution: 37 visited nodes: 8391 iterations: 8391 max fringe size: 733	total length of solution: 36 total cost of solution: 37 visited nodes: 8391 iterations: 8391 max fringe size: 733	total length of solution: 34 total cost of solution: 37 visited nodes: 7790 iterations: 7790 max fringe size: 925	total length of solution: 34 total cost of solution: 37 visited nodes: 5584 iterations: 5584 max fringe size: 890

Para analizar estos resultados hemos utilizado el ESCENARIO 1, y utilizando el algoritmo A-STAR hemos podido determinar que la última entrada de la tabla representa la mejor heurística. Se puede observar además que la heurística Manhattan es superior respecto a la que contempla únicamente el número de fotos, y dentro de esta también obtenemos mejores valores en la memoria y las iteraciones en la versión avanzada Manhattan 5 AVE.

ESCENARIO 1	ESCENARIO 2
Escenario de 7x10 casillas con 3 casillas de fotografías por hacer, dos puntos de recarga de pila y 6 casillas de agua	Escenario de 7x10 casillas y la ampliación a 4 casillas de fotografías por hacer, cuatro puntos de recarga de pila y 10 casillas de agua

Realizando la ejecución de estos dos escenarios para cada uno de los algoritmos de búsqueda obtenemos los resultados que se adjuntan en la tabla siguiente:

	Algoritmo	A-STAR *	BREADTH FIRST	ITERATIVE LIMITED DEPTH FIRST	UNIFORM COST
ESCENARIO 1	Heurística	total length of solution: 34 total cost of solution: 37 visited nodes: 5584 iterations: 5584 max fringe size: 890	total length of solution: 30 total cost of solution: 41 visited nodes: 5802 iterations: 5802 max fringe size: 549	total length of solution: 34 total cost of solution: 42 visited nodes: 49970 iterations: 49970 max fringe size: 84	total length of solution: 36 total cost of solution: 37 visited nodes: 8391 iterations: 8391 max fringe size: 733
ESCENARIO 2	Manhattan 5 Ave	total length of solution: 38 total cost of solution: 39 visited nodes: 17145 iterations: 17145 max fringe size: 2243	total length of solution: 36 total cost of solution: 43 visited nodes: 21626 iterations: 21626 max fringe size: 1387	total length of solution: 40 total cost of solution: 43 visited nodes: 144218 iterations: 144218 max fringe size: 99	total length of solution: 36 total cost of solution: 37 visited nodes: 20714 iterations: 20714 max fringe size: 1753

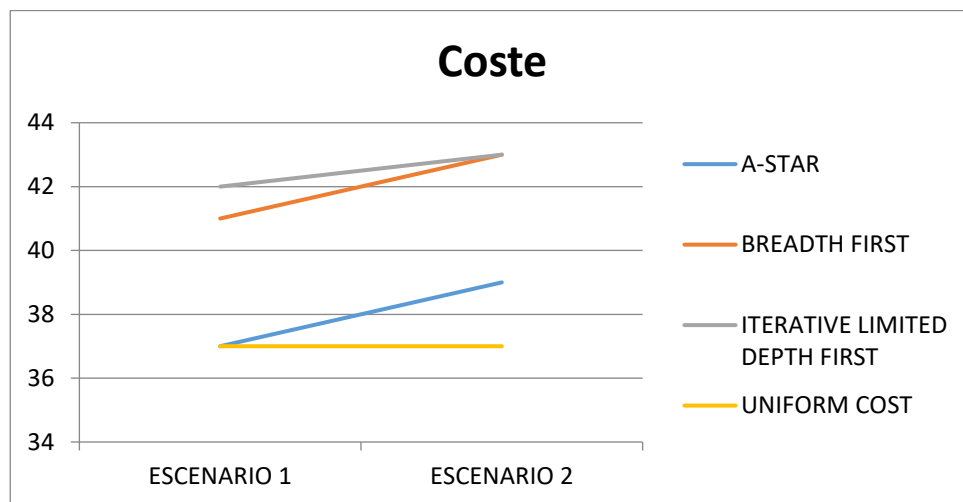
Como se puede observar en la tabla al ejecutar el juego, existe un aumento tanto de memoria, como de número de nodos visitados y el valor del coste del problema según se ha aumentado la complejidad del problema.

Por otro lado, si comparamos los algoritmos entre si podemos observar que, hay unas variaciones muy considerables del número de nodos visitados siendo el algoritmo de UNIFORM COST el más costoso a nivel de nodos y el menor ha sido el ITERATIVE LIMITED DEPTH FIRST en el caso del ESCENARIO 1 y el algoritmo BREADTH FIRST en el caso del ESCENARIO 2.

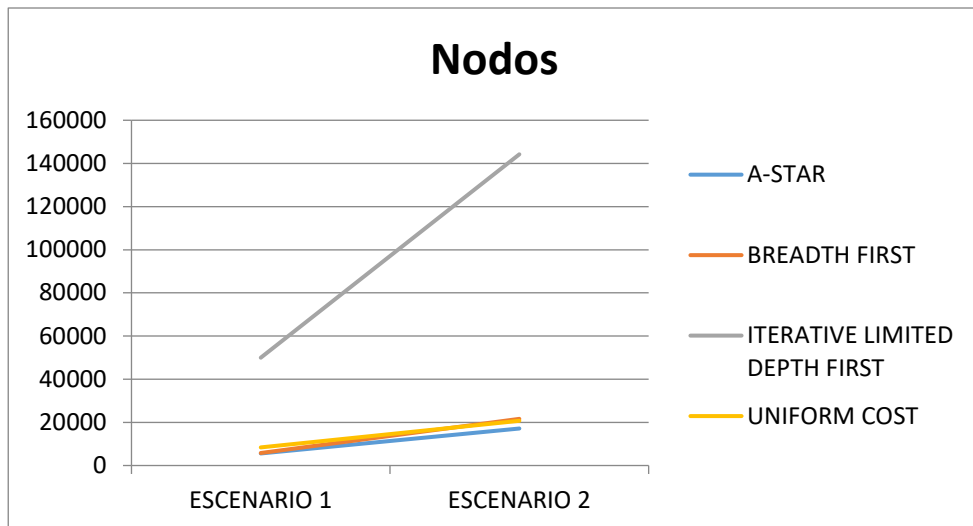
Además, a nivel de memoria el algoritmo cuya ejecución sería más óptima sería el ITERATIVE LIMITED DEPTH FIRST y siguiendo por aquel algoritmo cuya implementación sería la más costosa de memoria que sería la de A-STAR.

Finalmente, se analizará la ejecución de los algoritmos en función del coste en el cual destaca positivamente el algoritmo de UNIFORM COST, aunque todos los ellos tienen una media de un coste de 40, y ninguno de ellos se separa notoriamente del valor central de la media. Por último, cabe destacar que los algoritmos que destacan negativamente respecto al coste son BREADTH FIRST e ITERATIVE LIMITED DEPTH FIRST.

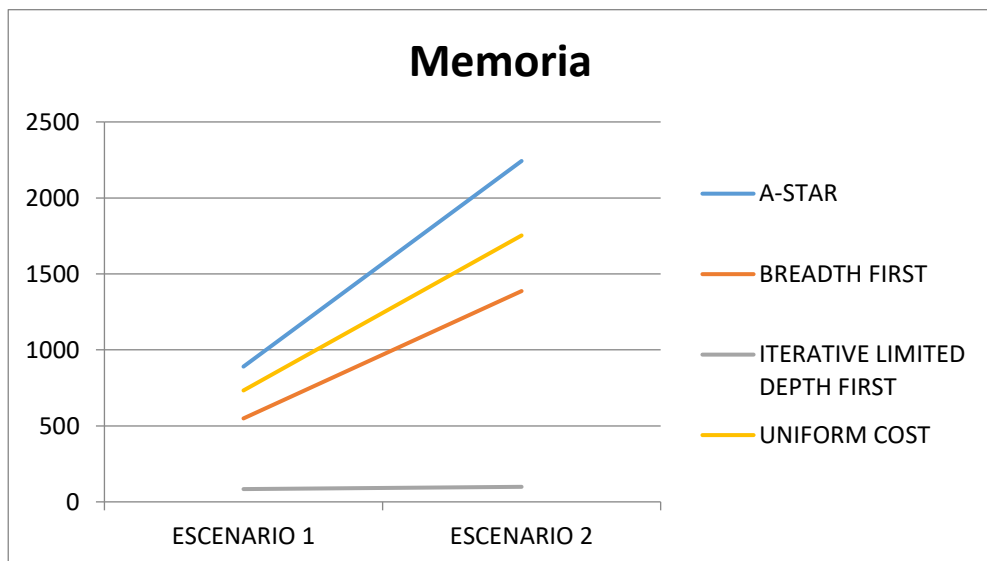
Como conclusión de este último análisis del experimento observamos que comparando todos los datos anteriores estos escenarios junto con las nuevas implementaciones de las nuevas funcionalidades del Dron, se detecta un aumento considerable de la complejidad y la eficiencia de los métodos debido a que todos los valores tanto de coste, como de memoria y número de nodos visitados han tenido un gran aumento en su ejecución.



Análisis: Respecto a este gráfico comparativo de los valores coste de cada escenario respecto de cada algoritmo de búsqueda observamos que todos ellos crecen linealmente con más o menos rapidez dependiendo del algoritmo excepto el de UNIFORM COST ya que utiliza unos costes uniformes para las casillas y por tanto no va a variar dependiendo de la complejidad del problema, además este es el algoritmo menos costoso. Por último cabe destacar, que en general el algoritmo más costoso en ejecución es el ITERATIVE LIMITED DEPTH FIRST.



Análisis: Respecto a este gráfico comparativo de los valores nodos visitados de cada escenario respecto de cada algoritmo de búsqueda observamos que todos ellos crecen linealmente con baja rapidez excepto el algoritmo ITERATIVE LIMITED DEPTH FIRST que crece más rápidamente. Finalmente cabe destacar que el peor algoritmo respecto a nodos es el ITERATIVE LIMITED DEPTH FIRST y el mejor es el algoritmo A-STAR.



Análisis: Respecto a este gráfico comparativo de los valores coste de memoria de cada escenario respecto de cada algoritmo de búsqueda observamos que todos ellos crecen linealmente con bastante rapidez dependiendo del algoritmo excepto el de ITERATIVE LIMITED DEPTH FIRST que es prácticamente constante comparado con los anteriores. Por último cabe destacar que el algoritmo que más costes de memoria requiere es A-STAR y el que menos ITERATIVE LIMITED DEPTH FIRST.

4. Conclusiones y comentarios

En esta práctica hemos aprendido como desarrollar diferentes tipos de búsqueda y analizar los resultados, además nos hemos familiarizado con el lenguaje Python y determinado heurísticas en busca de mejorar el problema de localización del dron. Hemos aprendido a realizar un análisis de resultados completo, tanto de memoria como de coste hasta al fin conseguir llegar a desarrollar un proyecto eficiente.

Siendo cierto que nos hubiera gustado extender los escenarios a dimensiones más grandes para poder establecer una diferenciación más marcada entre los escenarios, algoritmos y parámetros usados para analizar. Esto no ha sido posible dado que el espacio de búsqueda se volvía demasiado grande demorando demasiado el trabajo en el tiempo y no pudiendo ser factible.

En cuanto a la ejecución del programa queremos anotar que el tiempo del programa en la parte avanzada hasta encontrar la solución es algo extenso, no es inmediato debido a la amplia búsqueda y que hemos seleccionado un escenario amplio donde está reflejada la funcionalidad completa.

Anotación: dado que hemos realizado la práctica en dos partes, hemos introducido en la carpeta solución dos archivos, uno corresponde a la parte básica y el otro es la parte en la que se han añadido las funcionalidades avanzadas.