Universidad Carlos III

PRÁCTICA 2

HEURÍSTICA Y OPTIMIZACIÓN

ENTREGA: **14/12/2023**

GRUPO: **80**

Alumnos: **Ignacio Fernández Cañedo / Ana María Ortega Mateo**

INTRODUCCIÓN

En esta memoria se presenta un estudio tanto de satisfacción de restricciones como de búsqueda heurística. En la primera parte del problema, se consideran dos tipos de vehículos de transporte: Transporte Sanitario Urgente (TSU) y Transporte Individual no Urgente (TNU). Algunos de estos vehículos pueden llevar instalado un congelador. Estos vehículos deben estar aparcados en plazas especiales que disponen de conexión a la red eléctrica. El objetivo es asignar a cada vehículo una plaza de aparcamiento teniendo en cuenta varias restricciones, como el tipo de vehículo o la disponibilidad de conexión eléctrica en la plaza.

En la segunda parte, se presenta un mapa esquemático con la ubicación de los pacientes, los centros de atención y el parking del vehículo de transporte. Además, se añaden nuevas restricciones y consideraciones al problema, como la capacidad del vehículo o la necesidad de recargar la energía del vehículo. La función objetivo será la de minimizar el tiempo (coste energético) invertido en el transporte de todos los pacientes.

En ambos apartados, hemos modelado los problemas descritos y hemos implementado dicho modelo utilizando por ejemplo, un algoritmo de búsqueda heurística, específicamente el algoritmo A*.

Por último, se ha incluido en la memoria una sección en la que se analizan los resultados obtenidos, las restricciones utilizadas y se discuten las ventajas y desventajas de los enfoques utilizados, así como reflexiones personales acerca de la práctica.

APARTADO 1

1.1 Modelado

En el planteamiento del enunciado se pueden ver unas cuantas restricciones que afectan a nuestra función objetivo.

V: Conjunto de variables(ambulancias) = {ID-TNU-X,ID-TSU-X, ID-TNU-C, ID-TSU-C}

D(ID-TNU-X, ID-TSU-X): Dominio de cada variable sin congelador(todas las plazas) = {(1, 1), (1, 2), (1, 3)...}

D(TNU-C, TSU-C): Dominio de cada variable con congelador(plazas eléctricas) = {(1, 1), (1, 2), (2, 1)...}

El objetivo del primer apartado es obtener todas las asignaciones válidas de vehículos a plazas dentro de un parking.

Restricción 1 y 2: “Todo vehículo tiene que tener asignada una plaza y solo una.” , “Dos vehículos distintos, como es natural, no pueden ocupar la misma plaza”. Esta restricción se modeliza garantizando que la posición de un vehículo no pueda ser la misma que otro.

$$Vi \neq Vj \quad \forall i,j \in D$$

Restricción 3: “Los vehículos provistos de congelador solo pueden ocupar plazas con conexión a la red eléctrica.”. El dominio de los vehículos que disponen de congelador será únicamente el de las plazas eléctricas.

$$D(\text{TNU-C}, \text{TSU-C}) \in \{(1, 1), (1, 2), (2, 1)...\}$$

Restricción 4: “Un vehículo de tipo TSU no puede tener aparcado por delante, en su misma fila, a ningún otro vehículo excepto si este es también de tipo TSU. Por ejemplo, si un TSU está aparcado en la plaza 2.3 no podrá haber aparcado un TNU en las plazas 2.4, 2.5, 2.6....”. Para esto, establecemos una restricción para cada par de ambulancias de distinto tipo, en la que sí están en la misma fila, la posición de la ambulancia TNU deberá ser en alguna columna más atrás.

$$R_{\text{TNU-TSU}} = \{((x_i, y_i), (x_j, y_j))\} \qquad \text{si } x_i = x_j \text{ entonces } y_i < y_j$$

Restricción 5: “Por cuestiones de maniobrabilidad dentro del parking todo vehículo debe tener libre una plaza a izquierda o derecha (mirando en dirección a la salida). Por ejemplo, si un vehículo ocupa la plaza 3.3 no podrá tener aparcado un vehículo en la 2.3 y otro en la 4.3, al menos una de esas dos plazas deberá quedar libre”. En esta restricción, para cada trío de vehículos, si están en la misma columna, no pueden estar en filas adyacentes, garantizando así que siempre hay una plaza libre a la izquierda o a la derecha de cada vehículo.

$$Rvi\text{-}vj\text{-}vk = \{((xi, yi), (xj, yj), (xk, yk))\} \text{ si } yi = yj = yk \text{ entonces } |xi - xj| \neq 1 \wedge |xi - xk| \neq 1 \wedge |xj - xk| \neq 1$$

Por lo que nuestra modelización final sería:

Variables:

1) V: Conjunto de variables(ambulancias) = {TNU-X, TSU-X, TNU-C, TSU-C}

Dominios:

1) D(TNU-X, TSU-X): Dominio de cada variable sin congelador

2) D(TNU-C, TSU-C): Dominio de cada variable con congelador

Restricciones:

1) $Vi \neq Vj \qquad \forall i,j \in D$

2) $Vi \neq Vj \qquad \forall i,j \in D$

3) D(TNU-C, TSU-C) $\in$ plazas con conexión eléctrica

4) $R_{TNU-TSU} = \{((x_i, y_i), (x_j, y_j))\} \qquad$ si $x_i = x_j$ entonces $y_i < y_j$

5) Rvi-vj-vk = {((xi, yi), (xj, yj), (xk, yk))} si yi = yj = yk entonces |xi - xj| ≠ 1 ^ |xi - xk| ≠ 1 ^ |xj - xk| ≠ 1

APARTADO 2

2.1 Modelado

En este apartado se nos ha pedido encontrar una solución al problema, en caso de que la tenga, utilizando búsqueda heurística. Necesitamos trasladar a todos los usuarios del mapa a sus respectivos centros de tratamiento. Hay dos tipos de enfermos que son los no contagiosos(**N**) cuyo centro de tratamiento es CN y los contagiosos(**C**) cuyo centro de tratamiento será CC. Para resolver este apartado hemos realizado el algoritmo de búsqueda del A*. Para poder resolver bien este problema tenemos que tener en cuenta las distintas restricciones o limitaciones a la hora de resolverlo. Son las siguientes:

1. El vehículo tendrá que empezar en el parking y terminar en él.
2. El vehículo solo dispone de diez plazas, con dos plazas especiales. Las plazas especiales pueden estar ocupadas por algún usuario N pero si hay algún usuario no contagioso ocupando esas plazas especiales no se podrá subir ningún C. Los C solo pueden ocupar esas plazas especiales las otras no.
3. El vehículo tiene 50 unidades de energía, en caso de que esté quedando sin batería y no pueda realizar un camino deberá reponer su energía volviendo al parking y continuar con el proceso.
4. Los enfermos contagiosos serán los últimos en ser recogidos en los trayectos que intervengan y los primeros en ser dejados.
5. Siempre que se cumplan con las condiciones el paciente podrá subir al vehículo.
6. El vehículo solo se desplaza vertical y horizontalmete.

Un problema de búsqueda viene definido por sus estados, operadores y heurísticas, teniendo en cuenta el estado inicial y el final.

- **Estado**

Estado={posicion_mapa, energia, plazas no especiales,plazas especiales,no_contagiosos,contagiosos}

Para representar el estado en el que te encuentras hemos visto necesario saber en qué posición del mapa estás, con coordenadas x e y, la energía para saber si tienes que volver al parking y cuanta te queda, las plazas y plazas especiales para saber qué combinación de usuarios puedes realizar y la ocupación en sí del vehículo. Hemos reflejado el estado en el que te encuentras mediante nodos, de esta forma sabemos el coste que conlleva cada acción y por lo tanto sabemos cuánta energía se va consumiendo dependiendo en el nodo en el que te encuentres, y si el valor de las casillas es alguna en especial y las condiciones del problema lo permiten realizará la acción que tenga que hacer, cuando está en ese nodo en específico. También en cada estado se tiene en cuenta cuántos usuarios quedan por ser dejados, ya que para ser dejados tienen que haber sido recogidos. Si todos los usuarios son dejados a su destino correspondiente es lo que marca la diferencia entre el estado inicial y el final.

- **Estado inicial:**
  Estado_inicial = {parking, 50, [], [], num_nocontagiosos_mapa, num_contagiosos_mapa}

Nun_nocontagiosos_mapa hace referencia a todos los usuarios no contagiosos que hay en el mapa y no han sido recogidos y num_contagiosos_maoa a los contagiosos que no han sido recogidos.

- **Estado final:**
  Estado_final = {parking, 50, [], [], 0, 0}
- **Operadores**

En cuanto a los operadores hemos tenido en cuenta las posibles acciones que puede realizar el vehículo dependiendo del estado y situación que se encuentre:

1. **Encontrar_vecinos(nodo)**, encuentra a los vecinos en función de las casillas verticales y horizontales que están a distancia uno y si no son prohibidas(“X”).
2. **Distancia_Parking(nodo,energia,parking)**, tiene en cuenta la distancia a la que te encuentras del parking en el nodo en el que estás y si el vehículo no tiene la suficiente energía para ir a esos nodos del camino que se va siguiendo tendrá que volver. Recargando su energía a 50.
3. **Recoger_usuarioN(nodo, plazas, plazas especiales)**
4. **Recoger_usuarioC(nodo, plazas, plazas especiales)**
5. **Dejar_usuariosN(nodo,plazas,plazas especiales)**
6. **Dejar_usuariosC(nodo,plazas,plazas especiales)**

-Precondiciones:

3. if len(plazas+plazas_especiales)>=10 and num_nocontagiosos_mapa>0
4. if len(plazas_especiales)>=2 and not “N” in plazas_especiales and num_contagiosos_mapa>0
5. if (len(plazas)+len(plazas_especiales)==10 or (no_contagiosos==0 and contagiosos==0 and plazas!=[])) and not "C" in plazas_especiales
6. if (len(plazas_especiales)==2 or (contagiosos==0 and no_contagiosos==0 and plazas_especiales!=[])) and "C" in plazas_especiales and no_contagiosos==0

- **Heurísticas**

- **Heurística 1:** Se tiene en cuenta aquellas casillas que son especiales es decir, “P”, “C”, “N”,“CN” y “CC” y las heurística de los nodos que tengan esas posiciones será de 0 mientras

que si un nodo es vecino de otro nodo con valor especial su heurística valdrá uno, todas las demás valdrán 2. Esta heurística premia con menor valor a todas esas casillas(nodos) que tiene o bien de vecinos a una casilla relevante o lo son ellos mismos en el mapa. De esta forma expandirá antes los nodos relevantes.Esta heurística es admisible porque el coste real es mayor que la heurística. Básicamente, el propósito de esta heurística es guiar la búsqueda hacia los nodos más relevantes (las celdas especiales y sus vecinos) antes que los otros nodos. Esto puede ayudar a encontrar una solución de manera más eficiente, ya que se exploran primero los nodos que probablemente formen parte de la solución.

- **Heurística 2:** En esta heurística priorizamos el estado en el que se encuentra y el objetivo que quiere conseguir. Por ejemplo si tiene energía de sobra y puede recoger a algún usuario no contagioso, se priorizará estas casillas dándole el valor de dos a todas aquellas que no cumplen con ser "N" o "P" y los vecinos de estas casillas el valor de 1 y ellas mismas el valor de 0. Al parking independientemente de la situación en la que te encuentres le damos la heurística de 0 debido a que siempre viene bien que si está cerca de él se coloque para recargarse. Lo mismo pasa si hay hueco en las plazas especiales y no hay ningún no contagioso y hay energía suficiente para recoger a alguno de los no contagiosos, dándole 0 a los nodos con "C" o "P", uno a los vecinos y dos al resto. En caso de que las plazas están llenas o si está cerca de alguno de los centros, si cumple con las condiciones hará lo mismo siempre dejando antes a los C que a los N. Esta heurística es útil porque se adapta a las circunstancias cambiantes del problema y guía la búsqueda hacia las acciones más prometedoras en cada momento.

2.2 Implementación del código

Para realizar la resolución de este problema de búsqueda primero nos centramos en hacer el algoritmo de A*. Por ello, seguimos los siguientes pasos:

1. Nodo inicial I (estado inicial)
2. Definir la lista ABIERTA con el nodo inicial, la lista CERRADA vacía y ÉXITO como falso
3. Bucle que se repite hasta que 'ABIERTA' esté vacía o ÉXITO = Verdadero:
   a. Quitar el primer nodo de ABIERTA, N, y meterlo en CERRADA
   b. Si N es el Estado-Final → Entonces ÉXITO = Verdadero
   c. Si no, expandir N, generando el conjunto V de vecinos de N, que no son antecesores de N. De modo que:
      i. Se genera un nodo en G por cada nodo v del conjunto V
      ii. Se establece un puntero a N desde aquellos v de V que no estuvieran EXPANDIDOS
      iii. Se añaden los nodos v a ABIERTA
      iv. Para cada nodo v de V que estuvieran ya en ABIERTA o CERRADA se decide si redirigir o no sus punteros hacia N
      v. Para cada nodo v de V que estuviera en CERRADA se decide si redirigir o no los punteros de los nodos en sus subárboles
      vi. Se reordena ABIERTA en función de f(n) = g(n) + h(n)
4. Si EXITO = Verdadero → Entonces la solución será el camino desde I hasta N por los punteros: SOLUCIÓN = Camino
5. Si ÉXITO = Falso → Entonces no se encuentra solución: SOLUCIÓN = Fracaso

Para empezar leemos el fichero del mapa ***def leer_mapa*** y encontramos el nodo de partida que es el parking con la función ***def encontrar_parking***, creamos el nodo inicial con las condiciones del estado inicial con 50 de energía y con todos los usuarios por recoger. Y empezamos la búsqueda del A* con nuestra función ***def busqueda_a_estrella(mapa, nodo_inicial, nodo_objetivo, plazas, plazas_especiales ,caminos, ruta_final, contador_nodos_expandidos)***

Basándonos en esta idea de A* planteamos el problema con múltiples metas, haciendo un A* recursivo, dependiendo del estado en el que se encuentre y el objetivo según las condiciones en el que se encuentre el problema. En caso de que queden usuarios no contagiosos en el mapa y haya suficientes plazas tanto especiales como normales libres, la meta o el propósito será recoger a una "N", ya que primero tiene que recoger a las "N" antes que a las "C" . Cuando encuentre una "N", la restará del contador de los usuarios no contagiosos y dependiendo de si tiene suficiente energía para realizar el camino a la siguiente "N" o la siguiente meta, deberá volver al parking o seguir con su búsqueda encontrando el siguiente camino. Lo mismo pasa con las "C" en caso de que las plazas especiales no estén ocupadas por ninguna "N" y no haya más de dos "C" ocupando esas plazas, teniendo la suficiente energía, su meta será ir por una "C". En caso de que las plazas estén llenas y solo estén compuestas por "N" o sólo queden "N" en las plazas irá al centro de servicios "CN".

También si hay algún contagioso en el vehículo y hay que dejarle antes que a los no contagiosos su meta será "CC". Cuando haya terminado dejando a todos los usuarios en sus respectivos sitios volverá al parking, estableciendo ésta como meta final, ÉXITO FINAL. Cada vez que se cambia de meta llamamos al algoritmo A* para que calcule el camino en función de las heurísticas implementadas y siempre que se encuentra un camino se comprueba si tiene suficiente energía para volver al parking, ya que independientemente de la situación en la que se encuentre la meta final se encuentra en el parking si no tiene energía suficiente para volver no se puede continuar con ese camino y tiene que volver para recargarse porque no sería sino una solución válida. Esta comprobación la realizamos en la función ***def comprobar_distancia_parking(mapa, nodo_actual, nodo_objetivo,energia,caminos)*** en la que realiza el algoritmo de A* pero la meta fijada en el parking y teniendo en cuenta la energía que le va quedando en cada nodo. Si ese camino tiene una energía negativa cogerá la anterior ruta del parking.

Las heurísticas implementadas se realizan en la función ***encontrar_vecinos(nodo_actual, lista_existentes)*** donde asignará a un nodo la heurística(h), en función de las características de nuestra heurística. La clase Nodo está formada por los atributos de padre, siendo este su antecesor, posicionx del mapa, posiciones del mapa, coste(g), h en función de la heurística y del nodo que es y f qué es la suma de g y h. A la energía le vamos restando el coste de todos los nodos que hacen un camino y asegurando que hay suficiente para volver al parking. El nodo elegido al llamar a ***encontrar_vecinos*** irá en función de la que tenga menor f y el primer nodo que esté situado en la lista de abiertos. El camino de cada meta será establecido por los nodos padres que hacen que el nodo actual haya alcanzado el nodo objetivo(la meta). Las listas de abiertos y cerrados las vamos manejando con clases, y tenemos un contador que cuenta todos los nodos expandidos en total en todo el código, de todas las metas que va realizando. El código finalizará cuando se encuentre en el ESTADO FINAL y imprimirá con las funciones de impresión la información correspondiente pedida; def ***escribir_estadisticas*** **y** ***imprimir_camino_archivo*****.**

En definitiva, hemos implementado un enfoque integral para resolver el problema de búsqueda, utilizando un algoritmo A* adaptado a las condiciones particulares de la tarea. En lugar de un único

objetivo, optamos por un enfoque de múltiples metas, empleando recursividad en el algoritmo A* para abordar distintas metas según el estado del problema.

PARTE 3

3.1 Análisis de los casos de prueba empleados:

Parte-1

En la primera parte de la práctica, hemos implementado 4 casos de prueba distintos. En el primer caso, el archivo de entrada contiene la información indicada en el enunciado, una matriz 5x6 con plazas eléctricas en las posiciones (1,1) (1,2) (2,1) (4,1) (5,1) (5,2). En este caso hay 8 ambulancias, dos son del tipo TSU-C, otra del tipo TSU-X, otras dos TNU-C y el resto son TNU-X. En este caso podemos observar un mapa bastante amplio, con muchas posibles posiciones distintas para las ambulancias. Al ejecutar el archivo, el número de soluciones que se producen es de 2174288. A continuación de esto se muestran 3 soluciones escogidas de forma aleatoria entre el total. En todas estas soluciones podemos ver cómo se cumple con todas las restricciones, todas las ambulancias tienen espacio para maniobrar, y se respeta el orden de prioridades. A su vez también se respetan las plazas con conexión eléctrica, que únicamente contienen ambulancias con congelador. A continuación se muestra una posible salida del archivo:

	A	B	C	D	E	F
1	N. Sol: 2175288					
2						
3	Solución 1:					
4	–	1-TSU-C	–	5-TSU-X	–	–
5	7-TNU-C	–	–	–	–	–
6	–	3-TNU-X	–	–	6-TNU-X	–
7	8-TSU-C	–	–	–	–	–
8	–	4-TNU-C	–	–	–	2-TNU-X
9						
10						
11	Solución 2:					
12	–	4-TNU-C	–	–	–	–
13	7-TNU-C	–	–	–	3-TNU-X	–
14	–	–	2-TNU-X	6-TNU-X	5-TSU-X	–
15	–	–	–	–	–	–
16	8-TSU-C	1-TSU-C	–	–	–	–
17						
18						
19	Solución 3:					
20	1-TSU-C	8-TSU-C	–	–	5-TSU-X	–
21	–	–	–	2-TNU-X	–	–
22	3-TNU-X	–	–	–	–	6-TNU-X
23	4-TNU-C	–	–	–	–	–
24	–	7-TNU-C	–	–	–	–

La siguiente prueba consiste en un mapa 4x4 en el que podemos encontrar 5 plazas con conexión eléctrica. La característica principal de este mapa es que el número de ambulancias con conexión eléctrica es el número total de parkings con conexión eléctrica. Este número es de 5 unidades, y como podemos comprobar en la solución, los 5 parkings con conexión eléctrica están ocupados. También se han incluido otras 3 ambulancias adicionales sin congelador, y como se puede comprobar, todas las

soluciones respetan las restricciones de forma correcta. El número de soluciones en esta prueba es de 648. A continuación se muestra una posible salida del archivo:

	A	B	C	D	E	F
1	N. Sol: 648					
2						
3	Solución 1:					
4	2-TNU-C	1-TSU-C	5-TSU-C	–	–	–
5	–	–	–	6-TSU-X	–	–
6	–	–	–	–	–	–
7	3-TNU-C	4-TNU-C	8-TNU-X	7-TNU-X	–	–
8	–	–	–	–	–	–
9						
10						
11	Solución 2:					
12	4-TNU-C	2-TNU-C	3-TNU-C	6-TSU-X	–	–
13	–	–	–	–	–	–
14	–	–	8-TNU-X	7-TNU-X	–	–
15	5-TSU-C	1-TSU-C	–	–	–	–
16	–	–	–	–	–	–
17						
18						
19	Solución 3:					
20	2-TNU-C	5-TSU-C	1-TSU-C	–	–	–
21	–	–	–	6-TSU-X	–	–
22	–	–	–	–	–	–
23	3-TNU-C	4-TNU-C	8-TNU-X	7-TNU-X	–	–
24	–	–	–	–	–	–

En la siguiente prueba, hemos creado un mapa de dimensiones 3x3, con un número de 4 parkings con conexión eléctrica. El número de ambulancias es elevado, ya que lo que queremos comprobar con esta prueba es si se consiguen distribuir de forma correcta cuando tienen poco espacio. Tras ejecutarlo, podemos ver un número de soluciones bastante bajo, en concreto de 24. Este número de soluciones tan bajo muestra la correcta distribución de las ambulancias a lo largo del mapa, debido a la gran densidad de estas. Al observar las soluciones del problema, todas muestran como un carril siempre está vacío, asegurando así la maniobrabilidad de los vehículos en el parking. A continuación se muestra una posible salida del archivo:

	A	B	C	D	E	F
1	N. Sol: 24					
2						
3	Solución 1:					
4	6-TNU-C	4-TNU-C	7-TSU-X	–	–	–
5	–	–	–	–	–	–
6	3-TNU-X	2-TNU-X	5-TSU-X	–	–	–
7	1-TSU-C	–	–	–	–	–
8	–	–	–	–	–	–
9						
10						
11	Solución 2:					
12	4-TNU-C	6-TNU-C	5-TSU-X	–	–	–
13	–	–	–	–	–	–
14	3-TNU-X	2-TNU-X	7-TSU-X	–	–	–
15	1-TSU-C	–	–	–	–	–
16	–	–	–	–	–	–
17						
18						
19	Solución 3:					
20	4-TNU-C	6-TNU-C	7-TSU-X	–	–	–
21	–	–	–	–	–	–
22	3-TNU-X	2-TNU-X	5-TSU-X	–	–	–
23	1-TSU-C	–	–	–	–	–
24	–	–	–	–	–	–

Por último, se han incluido casos de prueba en los que no puede existir una solución debido a las restricciones empleadas. En uno de estos casos encontramos únicamente 3 parkings eléctricos, y a su

vez, 3 ambulancias con congelador. El principal problema y por lo que no hay solución es porque los parkings están juntos, no dejando así espacio a las ambulancias para maniobrar. Otro caso de prueba que hemos implementado en el que no se encuentra solución, es debido a que las plazas eléctricas situadas en la parte inferior del parking, al estar ocupadas en su totalidad por vehículos de tipo TSU, provocan que sea imposible cumplir con la restricción 4, que indica que delante de vehículos TSU no puede haber ningún vehículo TNU.

Parte-2

En la segunda parte de la práctica, se han implementado distintos casos de prueba para evaluar el comportamiento del código y asegurar su buen funcionamiento.
En el primer caso de prueba, hemos decidido obtener el mapa del ejemplo propuesto en el enunciado del ejercicio. En este ejemplo podemos observar como hay 11 'N' y 3 'C'. Una de las particularidades de este mapa es la distribución de dichos elementos, ya que estos se encuentran en algunas posiciones muy distantes. Para analizar el algoritmo y el funcionamiento de las heurísticas en profundidad, en primer lugar ejecutaremos el programa sin heurística, es decir, asignando la misma heurística a todos los nodos independientemente de su tipo para poder ver la mejora de las heurísticas implementadas. Tras hacer esto, vemos los siguientes resultados:

Tiempo total: 0.0140008522
Coste total: 180
Longitud del plan: 177
Nodos expandidos: 781

Como podemos observar, el coste total del camino no sobrepasa la longitud del plan, además de obtener un tiempo bastante bajo. La primera heurística que hemos decidido aplicar, como ya se ha comentado anteriormente, es aquella que reduce el valor heurístico de cuyos nodos tengan como vecino a un nodo con valor importante. Aplicando esta heurística obtenemos los siguientes resultados:

Tiempo total: 0.026006698
Coste total: 122
Longitud del plan: 113
Nodos expandidos: 556

Fijándonos en los valores obtenidos al aplicar dicha heurística, podemos observar una gran reducción de coste energético respecto al caso anterior. Esto se debe a que el algoritmo escoge de manera más inteligente los nodos hacia los que se debe dirigir. Como es obvio, el camino también se ha visto reducido cuantiosamente, junto al número total de nodos expandidos, ya que el problema está más informado y por lo tanto el resultado es más óptimo. Esta heurística se ve perjudicada en el aumento del tiempo respecto a la resolución anterior, aunque es un tiempo muy pequeño.
Siguiendo con las heurísticas propuestas, la siguiente heurística trata de priorizar el estado en el que se encuentra el problema para encontrar la solución más óptima. Dependiendo del estado en el que esté va a asignar a cada posición un determinado valor heurístico. Aplicando esta heurística obtenemos los siguientes resultados:

Tiempo total: 0.0505840778
Coste total: 173

Longitud del plan: 167
Nodos expandidos: 1241

Analizando el resultado obtenido tras aplicar esta heurística, observamos un poderoso aumento en el tiempo respecto a los anteriores resultados, aunque sigue siendo sumamente bajo. El coste total y la longitud del plano se han visto aumentados respecto a la prueba realizada con la heurística 1, pero sigue siendo inferior a la resolución de la prueba no informada. Esto implica que esta heurística no es tan eficiente como la anterior y a su vez, está menos informada. El aumento de tiempo y de nodos abiertos se debe a la cantidad de comprobaciones que se realizan, para verificar constantemente el estado en el que estamos y lo que se está buscando.
También se han realizado pruebas con otra heurística. Esta última heurística se trata de la distancia Manhattan, donde asignamos los valores heurísticos a los nodos en función de la distancia a la que están del propio parking. Muchas heurísticas se basan en una estimación de la distancia o el coste desde el estado actual hasta el estado objetivo. Esto puede hacer que la heurística sea muy específica para ciertos estados iniciales y finales, y no funcione tan bien para otros. El problema de esta heurística es que ante un problema como el propuesto, funcionará bien en el caso de que los objetivos estén próximos al nodo inicial, como veremos más adelante. En cuanto los nodos objetivos se alejen del parking, la heurística será peor que el propio caso no informado:

Tiempo total: 0.0290301276
Coste total: 202
Longitud del plan: 193
Nodos expandidos: 1290

Como se ha explicado previamente, esta heurística empleada únicamente para los casos de prueba, funciona peor que el resto para los casos en los que haya mucha separación entre los nodos.

El segundo caso de prueba empleado se trata de un mapa de dimensiones similares, en el que podemos encontrar el parking en una ubicación superior y un gran número de pacientes no contagiosos en la parte inferior. Al ejecutar esta prueba sin utilizar una heurística, es decir con h = 0 en todos los nodos ocurre lo siguiente:

Tiempo total: 0.018916815
Coste total: 191
Longitud del plan: 189
Nodos expandidos: 769

Observando estos datos, vemos unas proporciones similares a la prueba anterior. Aplicando nuevamente la primera heurística, suponemos que habrá una cuantiosa mejora tanto en el coste como en longitud del plan como en el número de nodos expandidos:

Tiempo total: 0.019646407
Coste total: 102
Longitud del plan: 97
Nodos expandidos: 462

Tal como predijimos, los valores mencionados se han visto considerablemente reducidos. En proporción a la prueba anterior, se han reducido bastante más. Esto implica que la heurística empleada

para este caso está muy informada y es muy útil para resolver el problema de forma óptima. Respecto a la segunda heurística utilizada, siguiendo con el resultado de la prueba anterior, intuimos una reducción en costes respecto a la heurística no informada, pero un aumento respecto a la anterior heurística:

Tiempo total: 0.046010494
Coste total: 156
Longitud del plan: 153
Nodos expandidos: 1174

Como era de esperar, el coste total se ha mantenido entre las dos pruebas anteriores, al igual que la longitud del plan, pero el número de nodos expandidos ha aumentado en número.
Como vimos en la anterior prueba, la heurística basada en el algoritmo de la distancia Manhattan no mejoraba en ninguno de los campos. En este caso, como los nodos objetivo están más cerca los unos de los otros respecto al parking, el algoritmo mejora en coste y en longitud del plan tanto a la heurística 2 como a la heurística no informada, pero expande más nodos al tener que verificar la distancia a todos los nodos objetivo:

Tiempo total: 0.0200176239
Coste total: 147
Longitud del plan: 141
Nodos expandidos: 1205

En la tercera prueba empleada, esta se trata de un ejemplo mucho más pequeño y sencillo, para poder verificar de manera más precisa todos los movimientos empleados por el algoritmo. Esta prueba cuenta con únicamente dos ‘N’ y una ‘C’, así como una casilla con coste 3 y otras con 2.
En este ejemplo, al utilizar el algoritmo directamente sin heurística, nos encontramos con un tiempo de ejecución extremadamente bajo:

Tiempo total: 0.000999689
Coste total: 28
Longitud del plan: 23
Nodos expandidos: 42

Como podemos ver en esta corta prueba, el camino está formado por 23 pasos, con un coste de 28, ya que cruza casillas con costes más elevados. Al aplicar la primera heurística, esperamos un decremento en el coste total y en la longitud del plan, ya que se espera que encuentre la solución más óptima posible:

Tiempo total: 0.00100405627
Coste total: 25
Longitud del plan: 19
Nodos expandidos: 40

El fichero de salida contiene los datos mencionados, en los cuales se ha acertado con la predicción. El coste total utilizado es menor ya que se ha optado por un camino más corto. Para la verificación del algoritmo, se ha seguido paso a paso el camino escogido, observando cómo se han tomado las

decisiones correctas. La ejecución de la segunda heurística muestra el siguiente archivo de salida:

Tiempo total: 0.015000169
Coste total: 28
Longitud del plan: 23
Nodos expandidos: 42

En este caso, esta segunda heurística no estaría relajando al problema, ya que al ser un mapa tan pequeño, con esta heurística se expande el mismo número de nodos que en el caso no informado, escogiendo el mismo camino. Al utilizar el algoritmo de la distancia Manhattan, observamos lo siguiente:

Tiempo total: 0.020011872
Coste total: 27
Longitud del plan: 19
Nodos expandidos: 54

La opción de utilizar el algoritmo de la distancia de Manhattan en esta prueba podría ser útil, ya que relaja en una parte el problema. Al ser un mapa pequeño, emplea bien la distancia a los nodos cercanos, aunque no por ello escoge el camino más óptimo.
La última prueba empleada se trata de un mapa de tamaño medio, en el que se han cambiado las posiciones de 'CN' y de 'P'. En este mapa, encontramos a 'CN' rodeado en una gran parte por zonas inaccesibles 'X' y a 'P' en una zona céntrica rodeada por casillas con coste 2 y por 'X'. También se han incluido usuarios contagiosos y no contagiosos en las esquinas del mapa, para intentar dificultar al algoritmo:

Tiempo total: 0.005008459
Coste total: 96
Longitud del plan: 95
Nodos expandidos: 311

En el caso de la implementación sin heurística, escoge el camino óptimo basándose en el coste. Al aplicar la primera heurística al problema, observamos lo siguiente:

Tiempo total: 0.007006847
Coste total: 82
Longitud del plan: 79
Nodos expandidos: 233

El coste se ha visto reducido junto a la longitud del plan. También se han expandido bastantes menos nodos que en el caso anterior. Veamos el resultado de la segunda heurística y si este se asemeja al de las anteriores pruebas:

Tiempo total: 0.0140154
Coste total: 83
Longitud del plan: 79
Nodos expandidos: 426

Como era de esperar, el resultado ha aumentado en la mayoría de aspectos respecto al caso anterior. A diferencia de en las otras pruebas, la diferencia del coste de esta no ha sido tan grande respecto a la primera heurística, aunque el número de nodos expandidos si se ha visto aumentado. Veamos como funciona en este caso la distancia Manhattan:

Tiempo total: 0.008012811
Coste total: 83
Longitud del plan: 81
Nodos expandidos: 348

En este caso, utilizar la heurística de la distancia Manhattan no sería conveniente, ya que como podemos ver no relaja en nada el problema.

En resumen, la elección de la heurística adecuada depende fuertemente de las características específicas del problema y del mapa. La primera heurística empleada demostró ser la más efectiva en general, ya que fue la que más relajó el problema en todos los casos, pero su desempeño puede variar en situaciones particulares. La segunda heurística ha demostrado ser útil en determinadas ocasiones, pero en ninguna mejor que la primera. Como hemos podido observar, la evaluación detallada de las heurísticas en diferentes contextos proporciona información valiosa para comprender cómo influyen en el rendimiento del algoritmo.

CONCLUSIONES

Este trabajo nos ha servido para comprobar y trabajar con la eficacia de los algoritmos de búsqueda y de la modelización con restricciones para resolver problemas de asignación complejos.Sin embargo, como con cualquier algoritmo de búsqueda, la eficacia puede variar dependiendo de las características específicas del problema y de la elección de la función heurística. Por lo tanto, es fundamental en la modelización, una heurística adecuada.

Una buena heurística puede guiar eficazmente la búsqueda hacia la solución, reduciendo el número de nodos que se deben explorar, el tiempo y el coste. Relajando de tal forma el problema y encontrando una solución de manera más eficaz. Muchas de las heurísticas dependen del estado inicial y del estado final para poder definirlas, teniendo que realizar un análisis difícil y pudiendo sacar heurísticas admisibles pero a lo mejor no las más óptimas.Una heurística puede funcionar bien para algunos casos pero no para otros. Esto puede ser especialmente cierto en problemas con una gran variabilidad en los estados iniciales y finales. Por lo que se podría definir una heurística específica para un caso en concreto del problema si sabemos que cumple con las condiciones que hacen que relaje el problema.

En cuanto a las modelización de satisfacibilidad de restricciones una vez modeladas y establecidas claramente y correctamente las restricciones es muy fácil saber si a la hora de implementarlo viendo todas las soluciones si los resultados cumplen con las soluciones o no, por lo que realizar análisis de los resultados en problemas de CSP es algo fundamental para saber si se sigue un correcto funcionamiento.

En resumen, la combinación de una heurística bien elegida y una modelización precisa de las restricciones puede conducir a soluciones robustas y eficientes para problemas de asignación complejos.