Universidad Carlos III
Curso Heurística y Optimización

Práctica
CSP y Búsqueda Heurística

FECHA: 15/12/23 ENTREGA: 2
GRUPO: 83 ID: 22

Alumno 1
Nombre: Alba Vidales Casado
Correo: 100472236@alumnos.uc3m.es
Alumno 2
Nombre: Martín Portugal Gonzalez
Correo: 100472279@alumnos.uc3m.es

1. Introducción **3**
2. Primera parte **3**
2.1. Modelo 3
Variables 3
Dominios 3
Restricciones 4
2.2. Análisis de los resultados 5
3. Segunda parte **6**
3.1. Modelo 6
Espacio de estados 6
Espacio de problemas 7
3.2. Análisis de heurísticas y resultados 8
Heurística 1 8
Heurística 2 8
Heurística 3 9
Heurística 4 9
4. Conclusión **12**

1. Introducción

En este documento se pueden observar las diferentes soluciones que proponemos para resolver los problemas mencionados en el enunciado del proyecto.

En primer lugar, tendremos la asignación de plazas de garaje dentro de los parkings para sus diferentes tipos de transportes, siguiendo para ello una serie de restricciones, como el tipo de ambulancia o la capacidad de que las plazas dispongan de una conexión eléctrica. Para esta parte, se requerirá extraer los datos de un archivo, al igual que devolverlos en otro. Y para resolverlo se utiliza la librería de python “constraint”.

Por otro lado, nos enfrentamos al desafío de diseñar el traslado de pacientes con el objetivo de minimizar el tiempo invertido en el transporte. Disponemos para ello de un mapa esquemático con la ubicación de los pacientes, centros de atención, parking y costes de tránsito, además de una serie de restricciones que limitan la forma en la que se puede realizar dicha tarea. Esto se hará con algoritmos de búsqueda y otras herramientas como las heurísticas, de las cuales posteriormente se analizará el rendimiento.

Por último, para ambas partes, se pide diseñar casos de prueba, para comprobar que el código funciona correctamente y analizar los resultados obtenidos.

2. Primera parte

2.1. Modelo

En la primera parte nos piden resolver un problema de restricción de satisfacción de restricciones en el cual tenemos encontrar la manera de colocar ambulancias en un parking con distintas limitaciones.

Para ello utilizamos un modelado CSP, lo primero que hay que hacer es definir la red de restricciones.

- R = (X ,D, C) → La red de restricciones ‘R’ va a estar formada por un conjunto de variables ‘X’. Estas van a estar definidas sobre su respectivo dominio ‘D’ que va a contener los posibles valores para cada variable, es decir, las posibles posiciones en el parking. Y por último, ‘C’ representará todo el conjunto de restricciones que limitan el problema.

Variables:

Cada una de las variables definidas en la red de restricciones van a representar una ambulancia diferente (están denotadas por el número que las identifique). Dichas ambulancias:

- Se identifican con números del 1 hasta el número total de ambulancias
- Pueden ser ambulancias no urgentes (TNU) o si urgentes (TSU)
- Disponen de congelador o no.

Es importante recalcar que se declararán tantas variables (ambulancias) como se especifique en el caso de uso que se esté probando, es decir, dependiendo de los valores que se reciban a través del documento de entrada.

Dominios:

Los dominios para las variables serán todas aquellas posiciones en las que se pueden situar. Dichas posiciones, van a estar limitadas por las dimensiones de dicho parking, que habrán sido seleccionadas según el input que reciba el programa. Además, dichos dominios estarán representados a través de tuplas, en las que el primer elemento representa la fila, y el segundo la columna $(filas, columnas)$.

En nuestro problema, inicialmente, podremos distinguir dos "variedades" de dominio principales para nuestras ambulancias:

- D_e {plazas con conexión eléctrica, que habrán sido indicadas en el input}
- D_t {todas las plazas del parking}

De esta manera, aquellas ambulancias que cuentan con un congelador, solo se podrán situar en las plazas que dispongan de electricidad, mientras que las ambulancias que no lo tengan, podrán situarse en cualquier plaza (incluyendo las eléctricas).

Restricciones:

Las restricciones, como ya sabemos, van a determinar las combinaciones que serán válidas para cada una de las posibles soluciones a nuestro problema.

Es importante destacar que en el enunciado se nos dice que "*Todo vehículo tiene que tener asignada una plaza y sólo una"*, lo cuál podría parecer una restricción. Sin embargo, no lo consideramos como una restricción, ya que evidentemente, para una solución dada, una ambulancia ha de tener un único valor asignado (que pertenezca a su dominio) y no más.

- **R1:** dos ambulancias diferentes no pueden ocupar una misma plaza, es decir, no pueden tener el mismo valor asignado. Para una ambulancia i y una ambulancia j se podría representar de la siguiente manera:

$$xi \neq xj, \quad \forall\, i \neq j$$

- **R2:** los vehículos que disponen de un congelador solo pueden ocupar plazas con conexión a la red eléctrica. Al modelar el problema, declaramos que las variables son las ambulancias y sus dominios los asignamos directamente en función de si tienen congelador o no, es por ello que no se modela como una restricción entre dos variables. Se limita al asignar el dominio.

- **R3:** Una ambulancia TSU no podrá tener aparcado delante ningún otro vehículo (es decir en la misma fila), a no ser que sea otra TSU. Por tanto, ninguna TNU estará aparcada delante de una TSU. Para una ambulancia $i \in TSU$ y otra ambulancia $j \in TNU$, donde f representa la fila y c representa el número de la columna, nuestra restricción Rij verificará que:

$$(fi,\ ci) \neq (fj,\ cj), \quad \forall\, i \neq j \quad \text{donde } cj\ = ci + n \text{ y } \ fj = fi$$

- **R4**: Todo vehículo debe tener libre una plaza a la izquierda o a la derecha. Para ello, si tenemos una ambulancia i, una ambulancia j, y una ambulancia k, la ambulancia i podrá ser adyacente con la ambulancia j siempre y cuando no haya una ambulancia k al otro lado de la ambulancia i.

$$\forall (i,\ j) \in \{(1,1), (1,2), \ldots (1,\ columnas)\ ,\ (filas,\ columnas)$$

$$[X(i\ +\ 1, j)\ \neq\ X(a, b)]\ \wedge\ [\,X(i\ -\ 1, j)\ \neq\ X(c, d)]$$

  En el caso de valores que se encuentren en los límites del recinto, es decir, en la primera o en la última fila. Entonces, si tenemos una ambulancia en la primera fila, tendremos que mirar que no haya ninguna otra ambulancia en la casilla inmediatamente inferior. Lo mismo pasaría con aquellas ambulancias situadas en la última fila, donde comprobaremos que no haya ninguna otra ambulancia en la posición inmediatamente superior.

2.2. Análisis de los resultados

Tras implementar nuestro código vamos a analizar los resultados obtenidos en los distintos mapas, realizando para ellos diferentes casos de prueba que irán verificando escenarios límites u otros aspectos que hemos considerado importantes. Para ver los inputs, acceder al documento *.txt* asociado a cada caso de prueba en la tabla inferior.

Archivo	Caso de prueba	Solución esperada	Solución obtenida
parking01.txt	Más ambulancias con congelador que plazas eléctricas.	No debe haber solución.	"N. Sol:",0
parking02.txt	Plazas eléctricas fuera del tamaño real del parking.	Esperamos una excepción y que no genere un .csv	“Las plazas eléctricas están fuera del rango”
parking03.txt	Mayor número de ambulancias que plazas en el parking.	Esperamos una excepción y que no se genere un .csv	“Hay más ambulancias que plazas disponibles”
parking04.txt	Comprobamos que las plazas eléctricas se asignan bien.	Dos soluciones posibles, las ambulancias en las dos plazas eléctricas intercambiadas.	"N. Sol:",2 "2-TNU-C","1-TNU-C" "–","–"
parking05.txt	Comprobamos que la TSU no tiene delante ninguna TNU.	Una única solución, con la TNU detrás de la TSU.	"N. Sol:",1 "–","–" "2-TNU-C","1-TSU-C"
parking06.txt	Comprobamos que no va a haber ambulancias adyacentes.	Ninguna solución, no se pueden poner todas las ambulancias porque incumple la restricción de adyacencia.	"N. Sol:",0
parking07.txt	Comprobamos que ambulancias sin congelador se pueden estacionar en plazas eléctricas.	Deberán salir 6 soluciones para un problema 2x2 con dos ambulancias.	"N. Sol:",6 "–","–" "2-TNU-X","1-TSU-X"
parking08.txt	Comprobamos que obtenemos una solución válida con un tamaño de parking mayor.	Deberá salir un gran número de soluciones, y mostrarnos una que cumpla todas las restricciones.	"N. Sol:",2175288 "–","7-TNU-C","3-TNU-X","6-TNU-X","–","–" "4-TNU-C","–","–","–","–","–" "–","–","–","–","–","–" "–","–","–","–","2-TNU-X","–" "8-TSU-C","1-TSU-C","–","–","–","5-TSU-X"

Para ejecutar todos los casos de prueba mencionados en la tabla anterior, habría que ejecutar el script que se ha incluido dentro del directorio *parte-1,* llamado *CSP-calls.sh*, en el cual se utiliza el comando ***python3*** para ejecutar todos estos archivos de prueba.

3. Segunda parte

3.1. Modelo

Espacio de estados

Cualquier problema de búsqueda, va a estar determinado por un espacio de estados. Dentro de este espacio de estados encontraremos dos componentes esenciales:

- **Estados:** los estados representarán a la ambulancia en un instante determinado de tiempo. Vamos a tener varios atributos dentro dichos estados para poder distinguirlos. Los vemos a continuación:
  - Posición $(i,\ j)$: i representa la posición en la fila y j es la posición en la columna.
  - Tipo: se debe saber en qué tipo de casilla se encuentra la ambulancia, este puede ser (C, N, CC, CN, P, X, 1,2, o cualquier otro número entero). Siendo C = domicilio de paciente contagioso, N = domicilio de paciente no contagioso, CC = centro de atención de paciente contagioso, CN = centro de atención de paciente no contagioso, P = parking, X = no transitable, 1, 2, 3, … = representado el coste energético de la casilla.
  - Pacientes a bordo: se debe saber que pacientes lleva la ambulancia en cada estado. Pueden ser contagiosos o no contagiosos. Habiendo un máximo de 8 plazas para no contagiosos y de 2 plazas para contagiosos.
  - Pacientes por recoger: se debe saber que pacientes quedan por recoger así como sus posiciones.
  - Energía: se debe saber cúal es la energía restante de la ambulancia.
  - g(n): se debe saber el coste de los arcos recorridos para llegar a ese estado.
  - h(n): representa el valor heurístico del estado, dependiendo de la función heurística utilizada.
  - Padre: representa el estado padre del que viene este estado.
  - Sucesores: representa los estados adyacentes de este estado.

- **Operadores:** los operadores van a ser las acciones que permitan pasar de un estado a otro. En este problema, vamos a tener los siguientes operadores:
  - Los desplazamientos de posición tendrán un coste asociado a g(n) y a la energía en función a la ubicación de la ambulancia para el estado al que se desplacen.
    - Desplazamiento izquierda: la ambulancia se desplaza una casilla hacia la izquierda, es decir su posición:
      $$(i,\ j) \rightarrow (i,\ j\ -\ 1)$$
    - Desplazamiento derecha: la ambulancia se desplaza una casilla hacia la derecha, es decir su posición:
      $$(i,\ j) \rightarrow (i,\ j)$$
    - Desplazamiento hacia arriba: la ambulancia se podrá desplazar a la casilla inmediatamente superior, de modo que:
      $$(i,\ j) \rightarrow (i\ -\ 1,\ j)$$
    - Desplazamiento hacia abajo: la ambulancia podrá desplazarse a la casilla inmediatamente inferior, de modo que:
      $$(i,\ j) \rightarrow (i\ +\ 1,\ j)$$
  - Recargar en el parking: si la ambulancia para por la casilla de parking, se recargará directamente la energía (E), esta acción no va a suponer un coste extra para g(n).
    $$E \rightarrow Emax$$

- Recoger pacientes contagiosos: si la ambulancia pasa por una casilla marcada como C, significa que es un domicilio de un paciente contagioso y la ambulancia puede recogerlo o no. Esta acción no supone un coste extra para g(n).

$$A\ bordo \rightarrow (no\ contagiosos,\ contagiosos\ +\ 1)$$
$$Por\ recoger \rightarrow (\ no\ contagiosos,\ contagiosos\ -\ contagioso(i,j))$$

- Recoger pacientes no contagiosos: si la ambulancia pasa por una casilla marcada como N, significa que es un domicilio de un paciente no contagioso y la ambulancia puede recogerlo o no. Esta acción no conlleva un coste extra para g(n).

$$A\ bordo \rightarrow (no\ contagiosos\ +\ 1,\ contagiosos)$$
$$Por\ recoger \rightarrow (\ no\ contagiosos\ -\ no\ contagiosos(i,j),\ contagiosos\ )$$

- Dejar pacientes contagiosos: si la ambulancia pasa por una casilla marcada como CC, significa que es un centro de atención para pacientes contagiosos donde la ambulancia dejará a dichos pacientes. Esto no supone un coste extra para g(n).

$$A\ bordo \rightarrow (no\ contagiosos\ ,\ 0)$$

- Dejar pacientes no contagiosos: si la ambulancia pasa por una casilla marcada como CN, significa que es un centro de atención para pacientes no contagiosos donde la ambulancia dejará a dichos pacientes. Esto no supone un coste extra para g(n).

$$A\ bordo \rightarrow (0,\ contagiosos)$$

Espacio de problemas

Los problemas de búsqueda también vienen determinados por un espacio de problemas. El espacio de problemas hace referencia al conjunto de situaciones o estados que derivan del estado inicial mediante un conjunto de operaciones. Dentro encontraremos cuatro componentes esenciales:

- **Conjunto de estados:** será el conjunto de todos los estados, es decir casillas, que conforman nuestro problema. El tamaño de este conjunto va a venir determinado por el archivo *.csv* que determinará el tamaño de nuestro tablero y por tanto el número de casillas que vamos a tener, así como la posición de cada una de estas casillas y el tipo de cada una de ellas.
  - Estado inicial: nuestro estado inicial va a ser la casilla del parking desde la que va a salir la ambulancia. Sus atributos estarán determinados por el archivo *.csv* que recibamos como entrada.
  - Estado final: nuestro estado final va a ser único, debe terminar en la casilla del parking, sin ningún paciente por recoger ni ningún paciente a bordo.
- **Conjunto de operadores:** va a ser el conjunto de todos los operadores que hemos descrito en el apartado anterior.

3.2. Análisis de heurísticas y resultados

En primer lugar, introduciremos el algoritmo usado para resolver los distintos mapas de recogida de pacientes.

El algoritmo A* estrella es un algoritmo de búsqueda heurística el cúal necesita de una función h(n) que de valor a los distintos estados. Este algoritmo expanda los nodos en función de su coste f(n), este f(n) es la suma de c(n) que son los costes de los arcos recorridos y de h(n) el valor heurístico del estado que depende de la función heurística utilizada. El funcionamiento es simple, se coge el estado con mejor función f de la lista abierta, si un estado tiene heurística = 0, significa que es el estado final y por tanto se ha encontrado la solución, si no se expandirá, se calcularán sus sucesores y se añadirán a la lista abierta por orden de f(n).

Ahora analizaremos y hablaremos de cada una de las heurísticas que hemos diseñado y que se utilizarán en el algoritmo A* implementado para la resolución del problema, las vemos:

Heurística 1:

Para obtener esta heurística hemos decidido relajar varias restricciones como pueden ser la energía utilizada (no la vamos a tener en cuenta), el número máximo de ocupantes que puede llevar la ambulancia y el orden de su recogida (no distinguiremos entre contagiosos y no contagiosos).

Para ello, en primer lugar calcula todas las distancias desde la posición actual hasta los diferentes domicilios en los que haya pacientes por recoger. Cuantos más pacientes quedan por recoger en dicho estado que queramos expandir, mayor será la heurística del mismo, favoreciendo el desplazamiento hacia aquellas casillas en las que haya pacientes por recoger.

A esto hay que sumarle la distancia a los centros de atención, en los cuales se penalizará la cantidad de pacientes a recoger de cada tipo, por ejemplo: si no quedan pacientes contagiosos por recoger, el valor de la heurística del estado del centro contagioso, va a ser más pequeño que en el caso de que no los haya recogido todavía. Priorizando que recoja pacientes antes de ir al centro.

Por último hay que considerar la distancia al parking, independientemente de si ya se han recogido todos los pacientes y se han dejado en los centros de atención. En caso de que suceda lo unico que cambiara sera que se pondrá a 0 la distancia a los parkings

Todo esto con el objetivo de generar una heurística que aunque poco informada, debido a que relajamos prácticamente todas las restricciones, nos diese un resultado que nos pudiese servir de guía con el resto de heurísticas y sus soluciones.

Heurística 2:

En esta otra heurística hemos decidido no relajar ninguna restricción y tener en cuenta todos los datos que se proponen en el problema.

Para conseguir lo anterior, hemos dividido el cálculo de la heurística en tres factores, el factor relacionado con los pacientes, el relacionado con los centro de atención y el relacionado con el parking.

Respecto a los pacientes, en esta heurística tendremos en cuenta el límite de pacientes por ambulancia, de 8 no contagiosos (10 en caso de no llevar ningún contagioso) y 2 contagiosos. Para ello hemos calculado la suma de los costes energéticos de los caminos desde la posición actual a los distintos domicilios, teniendo en cuenta que favorecemos que recoja pacientes no contagiosos antes. Y en el caso de llevar la máxima capacidad de pacientes no contagiosos, no se calcula el resto de pacientes no contagiosos sin recoger, ese valor será de 0.

Por supuesto, no se generarán estados en los cuales se supere las capacidades máximas de la carga de pacientes de la ambulancia así como no se recogerán pacientes no contagiosos si ya hay algún contagioso en la ambulancia.

En lo que respecta a los centros, se calcula la energía necesaria para ir desde la ambulancia hasta los centros de atención médica, favoreciendo que en caso de tener pacientes contagiosos a bordo se expandan aquellos estados que se acercan más a dicho centro de atención para este tipo de pacientes.

Por último, respecto del parking se calculará la energía necesaria para ir al mismo si ya se han recogido todos los pacientes y se han dejado en los centros de atención respectivos, en caso contrario se sumará una cantidad fija relacionada con la distancia máxima del parking.

Cabe destacar que el código no permite que se expandan nodos que dejen sin energía a la ambulancia, considerando volver al parking para recargar en caso de ser necesario.

Heurística 3:

Basándonos en la heurística 2 hemos decidido cambiar la restricción que involucra a la prioridad de los pacientes. De esta manera estamos obligando a la heurística a favorecer la recogida de aquellos pacientes contagiosos primero, pero cabe destacar que se mantiene la restricción de que tras subir a la ambulancia a un paciente contagioso no se puede subir a un paciente no contagioso.

Esta heurística se hizo con el objetivo de saber si el orden de recogida es un factor determinante para generar el camino óptimo o para conocer si la expansión de los nodos es mayor o menor.

Heurística 4:

Esta última heurística, solo va a relajar la restricción relacionada con la capacidad máxima de la ambulancia. De esta manera, no tendremos en cuenta a la hora de realizar el recorrido si se ha llegado o no a un máximo de pacientes que se encuentren dentro de la ambulancia (en la heurística 2 teníamos un máximo de 10).

Con ello, pudimos darnos cuenta de si el límite de ocupación de la ambulancia suponía una restricción importante a la hora de determinar el recorrido que realizaría nuestro algoritmo de búsqueda.

Conclusión heurísticas:

Podemos concluir haciendo un pequeño análisis de las heurísticas que acabamos de explicar. Por un lado, podremos decir que tanto la ***heurística 1*** como la ***heurística 4***, ambas creadas mediante la técnica de relajación de restricciones, van a ser heurísticas admisibles. Esto se debe a que el valor de dichas heurísticas para todos los estados, va a ser menor que la heurística real. Es decir:

$$h1(n) \leq h(n);\ h4(n) \leq h(n), \text{ donde además, } h1(n) \leq h4(n)$$

Por otro lado, tendremos la ***heurística 2*** y la ***heurística 3***. Por un lado, en la $h2$ se ha intentado conseguir una heurística lo más informada posible de acuerdo con el enunciado del problema, sin relajar ninguna restricción pero utilizando factores de multiplicación para orientar a la heurística de manera que encontrase el camino expandiendo menos nodos. Por otro lado, tendremos que $h3$ tampoco va a relajar ninguna restricción, sino que decidimos cambiar el orden de recogida para comprobar los efectos que tendría en la misma, y también utilizar factores de multiplicación para encontrar la solución de manera más rápida.

La utilización de dichos factores para estas dos últimas heurísticas, implicarán que estas sean inadmisibles, y que la solución que se encuentre, no tiene por qué ser la solución óptima del problema.

Resultados, casos de prueba

Mapa 01:

Para el *mapa01.csv* decidimos implementar un mapa 3x3 que será el mapa más básico de los implementados, pero con todos los tipos de casillas.

C	1	CC
1	CN	1
P	1	N

Para la ***heurística 1***, se expanden 352 nodos y la solución consta de 13 estados. Genera una heurística con muchos nodos consiguiendo una solución para un problema relajado.

La ***heurística 2*** Primero recoge al paciente no contagioso en (3,3) para posteriormente recoger al contagioso en (1,1). Esto respeta las restricciones del problema, además que deja primero al paciente contagioso y luego al no contagioso, finalizando el problema en el parking. Todo esto expandiendo 46 nodos y consiguiendo una solución en 13 nodos. Viendo que respecto a la heurística 1, aunque se expandan muchos menos nodos debido a que se infla la heurística, no se encuentra una mejor solución.

Para la ***heurística 3*** donde el orden está invertido, la heurística tarda menos ya que en vez de recorrer el mapa de N a C para luego ir a CC, directamente va a por el paciente contagioso y lo deja en su centro, siguiendo un camino distinto que le permite ahorrarse 2 nodos en la solución, generando un camino más corto de solo 11 nodos expandiendo 30.

Para terminar, la ***heurística 4*** al ser más informada que la primera heurística, es capaz de encontrar una mejor solución, con una longitud de 11 nodos, 2 menos que en la otra heurística admisible y expandiendo 126 nodos. Cabe destacar que la heurística 3 encuentra la misma solución expandiendo muchos menos nodos, pero al no ser admisible, no se puede tomar como una solución válida.

Mapa 02:

Para el *mapa02.csv* decidimos implementar un mapa 4x4 que añadirá casillas intransitables, casillas de coste 2 y más pacientes que recoger.

C	1	CC	C
1	CN	1	2
P	1	N	X
1	1	1	N

Para la ***heurística 1***, generará una solución de 17 estados con 13477 nodos expandidos, en la cual debido a que no tiene en cuenta el orden para recoger a los pacientes se dirigirá primero a los pacientes contagiosos ya que se encuentran más cerca. Para posteriormente dirigirse al CC. Esto tiene sentido para una heurística que no tenga en cuenta que los nodos de C se deben recoger los últimos, ya que sería más adecuado con toda la información haber recogido primero a N. Una vez hecho esto, se dirigirá a los pacientes no contagiosos y los dejará en su centro de atención.

La ***heurística 2***, genera una heurística con menor expansión de nodos pero una peor solución, teniendo esta una longitud de 19 nodos. Como era de esperar, primero se dirige a los pacientes no contagiosos para posteriormente recoger al C de la esquina superior derecha y de camino a recoger el otro contagioso dejará al primer contagioso en su centro, y posteriormente deja a los no contagiosos en su centro correspondiente.

Actúa como cabría esperar, primero los no contagiosos, luego los contagiosos y dependiendo de los costes a los centros, termina de recoger al resto de contagiosos o se dirigirá a los centros. Esto ejemplifica que una heurística no admisible, en este caso por usar factores de multiplicación sobre las distancias, no encuentra una mejor solución que la heurística 1.

La ***heurística 3***, genera una heurística con menor expansión y una solución de 17 nodos. Esto se debe a la disposición de los centros de atención y de los pacientes, en especial de los contagiosos. Al forzar que se recojan los contagiosos y debido a la distribución del mapa, se ahorran pasos para la solución, subiendo a la esquina superior derecha y desplazándose hasta el segundo contagioso y dejándolos en el centro correspondiente. Tras esto recoge los dos no contagiosos, los lleva a su

centro y vuelve al parking. En este caso en concreto vamos a tener menos pasos pero no tiene por qué ser así para otros mapas diferentes.

La ***heurística 4***, se trata de una heurística más informada que la 1, por tanto expande, como cabría de esperar, menos nodos (11834), y a su vez encontrando una mejor solución que consta de 17 nodos, 2 menos que para la heurística 1. Podemos apreciar que las heurísticas 2 y 3, aunque no admisibles son capaces de encontrar la misma solución.

Mapa 03:

Para el *mapa03.csv* decidimos implementar un mapa 3x3 donde agregamos costes más altos para ciertas casillas y así estudiar el comportamiento de la heurística.

C	1	CC
24	CN	1
P	20	N

Para la ***heurística 1***, como es de esperar, da la misma solución y la misma expansión de nodos que en el Mapa 01 anteriormente analizado, esto se debe a que esta heurística no tiene en cuenta los costes de energía de las casillas.

En la ***heurística 2***, el coste de la solución ha aumentado considerablemente, llegando a 50, mientras que la cantidad de nodos utilizados es solo de 13, 2 más que usando la primera heurística, y llegando a expandir 852 nodos. Si vemos el camino que ha seguido la solución nos damos cuenta de que primero recoge el paciente N, también favorecido por el camino energético, ya que es menor que si fuese a por el paciente C. Tras esto recoge el paciente C y posteriormente se dirige a los centros para dejar a los pacientes. Cabe destacar que tal y como está planteada la heurística, tiene prioridad recoger a los pacientes si todavía hay capacidad antes que pasar por los centros con pocos pacientes.

Con la ***heurística 3***, el coste del camino se mantiene pero varía la longitud del mismo aumentando hasta 13 y la expansión es mucho menor siendo solo de 112 nodos. Todo esto tiene una explicación, se debe al posicionar C, sigue un camino para recogerlo que aunque menos directo, es más eficiente en términos de energía. Lo que hace que solo yendo a C recorra casi la mitad del mapa, dejando que la expansión de los nodos se reduzca al dejar un camino mucho más claro en cuanto a los objetivos, ya que los pasos siguientes son dejar a C, recoger a N, dejarlo en CN y volver al parking. Pudiéndose realizar solo en ese orden.

Con la ***heurística 4***, a diferencia de anteriores mapas donde la primera heurística que era menos informada expandía más nodos, en este caso ocurre al revés. La heurística 4 expande muchos más nodos llegando a los 2323, debido a que la otra heurística admisible (la primera) no tiene en cuenta los costes energéticos, simplificando el problema y por tanto los nodos expandidos. También hay que tener en cuenta que encuentra la misma solución que la heurística 2, pero distinta de la 3 aunque de misma longitud, debido al orden de recogida de los pacientes. Como se ha mostrado en otros mapas las heurística 2 aunque no admisible, encuentra la misma solución que una heurística admisible.

Mapa 04:

Para el *mapa04.csv* decidimos implementar un mapa 4x4, pero con más contagiosos del límite permitido, para ver cómo se comportaba la heurística.

C	N	CC	C
1	CN	1	2
2	1	N	X
P	1	1	C

Para la ***heurística 1***, si lo comparamos con el Mapa 02, tiene sentido que se generen muchos más nodos ya que hay más pacientes por recoger, y la posición del parking se aleja un poco más que en el otro mapa, con un total de 3612 nodos expandidos y 25 nodos en la solución. Al ver la solución generada podemos comprobar que es una heurística poco informada, ya que prioriza el recoger pacientes habiendo maneras más eficaces de llegar a ellos. Por ejemplo, tras recoger el paciente de la casilla N en la posición (3,3) lo más eficiente sería recoger el paciente C de la posición (4,4), ya

que se encuentra muy alejado del resto y habría que volver a él desde la fila 1, que es a dónde se dirige según la heurística.

Para la ***heurística 2***, el coste respecto de la primera baja hasta 176 nodos expandidos pero encuentra una peor solución de 27 nodos. Si analizamos más en detenimiento el camino podemos ver que primero se recogerá a los pacientes N, para posteriormente recoger a los C y dejarlos en el orden pedido. Aunque esto pueda parecer correcto, no es lo más eficiente, ya que igual que en la heurística 1 cuando se dirige al N más cercano (3,3) no recoge el C de (4,4), penalizando el camino. Por otro lado, esta heurística al recoger el segundo N, decide ir a por el C del (1,4) en vez del (1,1).

Para la ***heurística 3***, mantiene el camino de 26 y los nodos expandidos a 94, esto se debe a que al forzar a recoger primero los pacientes contagiosos, y no permitir que se suban los pacientes no contagiosos, trayectos en los que podrías recoger un N y un C juntos, por proximidad, se conviertan en 2. Esto encarece inevitablemente los costes del camino. Si por ejemplo nos hubiésemos encontrado un mapa mucho más grande como un 20x20 con contagiosos en las esquinas, nos encontraríamos ante una heurística terriblemente ineficiente.

La ***heurística 4*** aunque nos da la misma solución que la heurística 2, expande muchos más nodos. Llegando a los 48206, Esto se debe a que en un principio se pueda pensar que el camino sería menor ya que tiene más capacidad, al encontrarse los pacientes tan cerca de los centros de atención, la capacidad no es un factor decisivo para encontrar la solución óptima. Sin embargo, generará más nodos al tener más opciones en cuanto a la capacidad de llevar pacientes se refiere.

4. Conclusión

La realización de este proyecto nos ha proporcionado una inmersión total en la resolución de problemas a través de dos enfoques diferentes: la validación con Python Constraint de una red de restricciones por un lado, y la planificación heurística con el algoritmo de búsqueda A* por el otro. Ambos problemas,nos han permitido aplicar conceptos teóricos que hemos aprendido en la asignatura de manera práctica.

En la primera parte, al abordar la validación con Python Constraint, pudimos observar la importancia de modelar adecuadamente el problema como un CSP, y la repercusión de hacerlo mal. La utilización de la librería python-constraint ha resultado ser una herramienta eficiente, y que además nos facilitó la resolución del problema y obtención de las soluciones, simplificando la representación de restricciones.

En el segundo apartado, al trabajar con búsqueda heurística utilizando el algoritmo A*, pudimos estudiar la implementación de técnicas avanzadas que nos permitirían optimizar la búsqueda de soluciones en un espacio de estados extenso. El diseño y evaluación de las diferentes heurísticas empleadas en dicho algoritmo de búsqueda, nos han ayudado a comprender cómo elegir funciones de estimación adecuadas, y cómo dicha elección puede impactar directamente en el rendimiento del algoritmo.

En su conjunto, la práctica nos ha proporcionado una visión completa sobre la aplicación de diferentes técnicas estudiadas en la asignatura para resolver problemas del mundo real. Desde la fase de modelado y la definición de restricciones hasta la programación de algoritmos de búsqueda. Este proyecto, por tanto, permitió no solo comprender los fundamentos teóricos, sino también aplicarlos de manera efectiva en situaciones prácticas.