Práctica 2: *Satisfacción de Restricciones y Búsqueda Heurística*

Heurística y Optimización.

Grado de Ingeniería en Informática. Curso 2023-2024

Planning and Learning Group
Departamento de Informática

1. Objetivo

El objetivo de esta práctica es que el alumno aprenda a modelar y resolver problemas tanto de satisfacción de restricciones como de búsqueda heurística.

2. Enunciado del problema

Los responsables del servicio de ambulancias han quedado satisfechos con el trabajo que hemos realizado y se han puesto en contacto con nosotros para que diseñemos la asignación de sus diferentes tipos de transportes a las plazas de garaje dentro de los *parkings*. En concreto disponen de dos tipos de transporte:

- Transporte Sanitario Urgente (*TSU*), atendido por ambulancias totalmente medicalizadas.
- Transporte Individual no Urgente (*TNU*), que atiende los traslados de los pacientes que no necesitan de una asistencia médica durante el trayecto.

Además, cualquiera de los dos tipos de transporte puede llevar instalado, o no, un congelador especial para el trasporte de muestras y medicamentos que requieren muy baja temperatura para su conservación. Dichos vehículos deben estar aparcados, dentro de los *parkings*, en unas plazas especiales que disponen de conexión a la red eléctrica. Un esquema de una posible configuración de un *parking* puede observarse en la Tabla 1:

De dicha tabla podemos extraer la siguiente información:

- Los aparcamientos están numerados por fila y columna.
- Los aparcamientos marcados en verde son plazas especiales con conexión a la red eléctrica.
- En marrón, y con el símbolo >>, se ha marcado la ubicación de la salida del garaje.

1,1	1,2	1,3	1,4	1,5	1,6	>>
2,1	2,2	2,3	2,4	2,5	2,6	>>
3,1	3,2	3,3	3,4	3,5	3,6	>>
4,1	4,2	4,3	4,4	4,5	4,6	>>
5,1	5,2	5,3	5,4	5,5	5,6	>>

Tabla 1: Esquema de un *parking*.

2.1. Parte 1: Validación con Python constraint

Para atender a las diferentes necesidades de ubicación de los vehículos dentro del *parking* debemos tener en cuenta lo siguiente:

1. Todo vehículo tiene que tener asignada una plaza y sólo una.
2. Dos vehículos distintos, como es natural, no pueden ocupar la misma plaza.
3. Los vehículos provistos de congelador sólo pueden ocupar plazas con conexión a la red eléctrica.
4. Un vehículo de tipo *TSU* no puede tener aparcado por delante, en su misma fila, a ningún otro vehículo excepto si éste 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*...
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.

Para esta parte se pide:

1. Modelar este problema como un problema de satisfacción de restricciones CSP de modo que se puedan obtener asignaciones válidas de vehículos a plazas dentro de un *parking*.
2. Utilizando la librería *python-constraint*, desarrollar un programa que codifique dicho modelado de modo que el problema pueda representarse y resolverse con dicho programa.

El programa desarrollado recibirá como entrada un fichero, con toda la información necesaria para resolver el problema, sobre el *parking* y vehículos. Tendréis que generar vuestros propios casos de prueba, con el fin de analizar el comportamiento del programa, siguiendo el siguiente ejemplo:

```
5x6
PE:(1,1)(1,2)(2,1)(4,1)(5,1)(5,2)
1-TSU-C
2-TNU-X
3-TNU-X
4-TNU-C
5-TSU-X
6-TNU-X
7-TNU-C
8-TSU-C
...
```

Estructura del fichero:

```
1ª línea: filas x columnas del parking
2ª línea: listado de plazas con conexión eléctrica
3ª y posteriores: código de identificación de los
   vehículos [ID-TIPO-CONGELADOR]
       ID: nº secuencial
       TIPO: TSU/TNU (Urgente/No urgente)
       CONGELADOR: C/X (Con congelador/Sin congelador)
```

El programa deberá ejecutarse desde una consola o terminal con el siguiente comando:

```
python CSPParking.py <path parking>
```

Donde `path parking` define la ruta donde se encuentra el fichero de entrada correspondiente a los datos del parking.

El programa debe generar un fichero de salida en el mismo directorio donde se encuentran los ficheros de entrada. El nombre del fichero de salida será de la forma `<parking>.csv` (**texto entrecomillado separado por comas**). Por ejemplo, si el fichero utilizado en la llamada es `parking01`, el nombre del fichero de salida será `parking01.csv`.

El fichero de salida tendrá en la primera línea el número de soluciones encontradas y, a continuación, al menos una solución del problema (aunque es deseable que muestre algunas más de forma aleatoria). Un posible ejemplo del contenido del fichero de salida, que de hecho es una solución del ejemplo propuesto, sería el siguiente:

```
"N. Sol:",302
"7-TNU-C","1-TSU-C","5-TSU-X","-","-","-"
"-","-","-","-","-","-"
"2-TNU-X","3-TNU-X","6-TNU-X","-","-","-"
"-","-","-","-","-","-"
"4-TNU-C","8-TSU-C","-","-","-","-"
```

Donde:

1. La primera línea indica el número de soluciones encontradas.
2. Las siguientes líneas representan a las filas de apartamientos del *parking*
   a) ``-'' si la plaza no es ocupada.
   b) ``ID-TIPO-CONGELADOR'' (con la estructura vista anteriormente) si la plaza es ocupada.
3. Al importar el fichero en una hoja de cálculo se tiene que poder observar sin problemas la estructura del *parking* con su ocupación.

Las llamadas al programa para ejecutar los casos de prueba que diseñe el alumno se deben incluir en un script llamado `CSP-calls.sh`. Un posible ejemplo del contenido de este script es el siguiente:

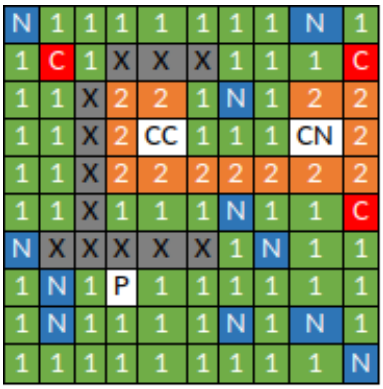
```
python CSPParking.py ./CSP-tests/parking01
python CSPParking.py ./CSP-tests/parking02
...
```

2.2. Parte 2: Planificación con Búsqueda Heurística

El servicio de urgencias atiende, además, el traslado de usuarios a sus respectivos centros de tratamiento. Tenemos dos tipos de usuarios: con enfermedades infectocontagiosas y sin ellas. Para poder diseñar dicho traslado se nos proporciona un mapa, esquemático, como el mostrado en la Figura 1. El servicio se realizará en un transporte eléctrico colectivo.

En dicho mapa podemos observar la siguiente información:

- Celda marcada con una *N*: Domicilio de paciente sin enfermedades infectocontagiosas.
- Celda marcada con una *C*: Domicilio de paciente con enfermedades infectocontagiosas.
- Celda marcada con una *CC*: Centro de atención a pacientes con enfermedades infectocontagiosas.
- Celda marcada con una *CN*: Centro de atención a pacientes sin enfermedades infectocontagiosas.
- Celda marcada con una *P*: Situación del *parking*.
- Celda marcada con un número (**1, 2, 3...**): tiempo (coste energético) para transitar por esa ubicación. El tiempo/coste necesario para transitar por celdas sin valor (*N*, *C*, *CC*, *CN* o *P*) será de una unidad.



1	Tiempo/Coste energético)
2	
N	Usuario no contagioso
C	Usuario contagioso
CC	Atención enf. Contagiosas
CN	Atención enf. no contagiosas
P	Parking
X	No transitable

Figura 1: Mapa esquemático con la ubicación de pacientes, centros de atención, *parking* y tiempos de tránsito.

- Celda marcada con una *X*: posiciones no transitables.

Para realizar esta tarea se dispone de un vehículo colectivo que, por compromiso medioambiental, es eléctrico. El transporte podrá realizar tantos trayectos como sea necesario y tiene las siguientes características:

- Saldrá del *parking* y terminará en él. Puede recoger, sin superar la capacidad del vehículo, a tantos pacientes como sea necesario en un sólo trayecto para llevarlos a sus respectivos centros de atención.
- El vehículo dispone de 10 plazas en total, con 2 dos de ellas especialmente habilitadas para pacientes contagiosos. Los pacientes no contagiosos podrán usar también estas dos plazas siempre y cuando no se trasladen en ningún momento enfermos contagiosos en su mismo viaje.
- El vehículo tiene 50 unidades de energía a carga máxima. Utiliza tantas unidades de energía como se indique en cada celda del mapa (como ya se ha comentado anteriormente) y, si no fuera suficiente con una carga de 50 unidades para realizar todos los trayectos necesarios, deberá pasar por el *parking* para recargar. Esta acción de recarga es inmediata y no tiene un coste adicional.
- Los enfermos contagiosos son los últimos en ser recogidos de su domicilio y los primeros en ser dejados en sus respectivos centros de atención en cada trayecto en los que intervengan.
- Siempre que el vehículo transite por el domicilio de un paciente, y las condiciones permitan recogerlo, el paciente subirá al vehículo.
- El vehículo sólo se puede trasladar sobre al mapa a celdas adyacentes horizontal o verticalmente.

En esta parte se pide:

1. Modelar el problema de traslado de pacientes como un problema de búsqueda con el objetivo de minimizar el tiempo (coste energético) invertido en el transporte de todos los pacientes.
2. Implementar el algoritmo A* (en el lenguaje de programación que se considere) que resuelve el problema.
3. Diseñar e implementar funciones heurísticas (al menos 2), que estimen el coste restante, de modo que sirvan de guía para los algoritmos de búsqueda heurística.
4. Proponer casos de prueba y resolverlos con la implementación desarrollada. Estos casos se deben generar razonablemente dependiendo de la eficiencia alcanzada en la implementación.
5. Realizar un análisis comparativo del rendimiento del algoritmo con las heurísticas implementadas.

La implementación deberá recibir como parámetros de entrada:

1. La dirección del fichero de *mapa.csv* con toda la información sobre la ubicación de los pacientes, centros de atención, *parking*, etc. Ese fichero, *mapa.csv*, tendrá el siguiente formato (que se corresponde con el mostrado en la Figura 1):

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

2. Un parámetro (*num-h*) para determinar la heurística que debe utilizar la implementación (***1*** para la primera heurística implementada y ***2*** para la segunda).

Los casos de prueba se tendrán que poder ejecutar desde una consola o terminal con el siguiente comando:

```
python ASTARTraslados.py <path mapa.csv> <num-h>
```

Donde, `ASTARTraslados.py` es el script que permite invocar al programa desarrollado, `path mapa.csv` define el mapa con todos los elementos importantes a tener en cuenta y `num-h` es el anteriormente comentado.

Al igual que en la primera parte, las llamadas al programa para ejecutar los casos de prueba que diseñe el alumno se deben incluir en un script adicional llamado `ASTAR-calls.sh`, cuyo contenido será de la forma:

```
python ASTARTraslados.py ./ASTAR-tests/mapa.csv 1
python ASTARTraslados.py ./ASTAR-tests/mapa.csv 2
python ASTARTraslados.py ./ASTAR-tests/mapa02.csv 1
python ASTARTraslados.py ./ASTAR-tests/mapa02.csv 2
...
```

Donde `./ASTAR-tests` es la dirección de los ficheros de los casos de prueba. De esta manera se podrán ejecutar varios casos de prueba a la vez.

El programa debe generar dos ficheros de salida. Ambos se deben generar en el mismo directorio donde se encuentran los ficheros de entrada:

- Fichero con la solución del problema. Debe contener una línea por casilla transitada con el siguiente formato: `(x,y):valor:carga`, donde `x` es el número de fila en el mapa, `y` el de columna, `valor` el coste o tipo de celda (*N*, *C*, *CC*, *CN* o *P*) según el caso y `carga` el valor de carga que le queda al vehículo en ese momento. Por ejemplo:

```
(8,4):P:50
(8,3):1:49
(8,2):N:48
(9,2):N:47
(9,1):1:46
(8,1):1:45
(7,1):N:44
...
```

El nombre del fichero será de la forma `<mapa>-<num-h>.output` donde `<num-h>` es el número de la heurística utilizada en la ejecución. Por ejemplo para la primera línea de la llamada anterior el fichero con la solución se denominará `mapa-1.output` y se encontrará en el directorio `./ASTAR-tests`.

- Fichero de estadísticas. Este fichero debe contener información relativa al proceso de búsqueda, como el tiempo total, coste total, longitud de la solución y los nodos expandidos. Por ejemplo,

```
Tiempo total: 145
Coste total: 38
Longitud del plan: 27
Nodos expandidos: 132
```

Donde:

- `Tiempo total` es el tiempo que ha tardado el programa implementado en encontrar la solución.
- `Coste total` es el coste energético (tiempo) de la solución encontrada.
- `Longitud del plan` es el número de nodos desde el inicial hasta la solución.
- `Nodos expandidos` es el total de nodos expandidos por el algoritmo.

En este caso el fichero tendrá extensión `.stat`: `<mapa>-<num-h>.stat`.

3. Desarrollo de la práctica

La práctica se realizará en grupos de dos estudiantes. Se utilizará Github Campus Education, por lo que es indispensable que ambos estudiantes estén dados de alta.

4. Directrices para la Memoria

La memoria debe entregarse en formato pdf y tener un máximo de 15 hojas en total, incluyendo la portada, el índice y la contraportada. Al menos, ha de contener:

1. Breve introducción explicando los contenidos del documento.
2. Descripción de los modelos, argumentando las decisiones tomadas.
3. Análisis de los resultados.
4. Conclusiones acerca de la práctica.

Importante: Las memorias en un formato diferente a pdf serán penalizadas con 1 punto.

La memoria **no debe incluir código fuente** en ningún caso.

5. Evaluación

La evaluación de la práctica se realizará sobre 10 puntos. Para que la práctica sea evaluada deberá realizarse al menos la primera parte de la práctica:

1. Parte 1 (4 puntos)
   - Modelización del problema (1 punto).
   - Implementación del modelo (2 puntos).

- Resolución y análisis de los casos de prueba (1 punto).

2. Parte 2 (6 puntos)
   - Modelización del problema (1 puntos).
   - Implementación del algoritmo (3 puntos).
   - Resolución de casos de prueba y análisis comparativo de las heurísticas implementadas (2 puntos).

En la evaluación de la modelización del problema, un modelo correcto supondrá la mitad de los puntos. Para obtenerse el resto de puntos, la modelización del problema deberá:

- Ser formalizada correctamente en la memoria.
- Ser, preferiblemente, sencilla y concisa.
- Estar bien explicada (ha de quedar clara cuál es la utilidad de cada variable/restricción).
- Justificarse en la memoria todas las decisiones de diseño tomadas.

En la evaluación de la implementación del modelo, un modelo correcto supondrá la mitad de los puntos. Para obtenerse el resto de puntos, la implementación del problema deberá:

- Corresponder íntegramente con el modelo propuesto en la memoria.
- Entregar código fuente correctamente organizado y comentado. Los nombres deben ser descriptivos. Deberán añadirse comentarios en los casos en que sea necesario para comprenderlo.
- Contener casos de prueba que muestren diversidad para la validación de la implementación.

Se comprobarán los push al repositorio y que ambos miembros del equipo han contribuído.

6. Entrega

Se tiene de plazo para entregar la práctica hasta el 15 de diciembre de 2023 a las 23:59.
Sólo un miembro de cada pareja de estudiantes debe subir:

- Un único fichero `.zip` a la sección de prácticas de Aula Global denominado *Entrega Práctica 2*.
- El fichero debe nombrarse `p2-NIA1-NIA2.zip`, donde `NIA1` y `NIA2` son los últimos 6 dígitos del NIA (rellenando con 0s por la izquierda si fuera preciso) de cada miembro de la pareja. Ejemplo: `p2-054000-671342.zip`.
- La memoria en formato pdf debe entregarse a través del enlace Turnitin denominado *Entrega Memoria Práctica 2*. La memoria debe entregarse en formato pdf y debe llamarse `NIA1-NIA2.pdf` — después de sustituir convenientemente el NIA de cada estudiante. Ejemplo: `054000-671342.pdf`

La descompresión del fichero entregado en primer lugar debe generar un directorio llamado `p2-NIA1-NIA2`, donde `NIA1` y `NIA2` son los últimos 6 dígitos del NIA (rellenando con 0s por la izquierda si fuera preciso) de cada miembro de la pareja. Este directorio debe contener: primero, la misma memoria en formato pdf que ha sido entregada a través de Turnitin, y debe llamarse `NIA1-NIA2.pdf` — después de sustituir convenientemente el NIA de cada estudiante; segundo, un fichero llamado *autores.txt* que identifique a cada autor de la practica en cada línea con el formato: NIA Apellidos, Nombre. Por ejemplo:

```
054000 Von Neumann, John
671342 Turing, Alan
```

La descompresión de este fichero debe producir al menos dos directorios llamados exactamente "`parte-1`" y "`parte-2`", que contengan los archivos necesarios para ejecutar correctamente cada una de las partes.

Importante: no seguir las normas de entrega puede suponer una pérdida de hasta 1 punto en la calificación final de la práctica.

Se muestra a continuación una distribución posible de los ficheros que resultan de la descompresión: