Heurística y Optimización

Práctica 2

Arturo Cardenal López-Santos
100432160

Repositorio: https://github.com/100432160/heuristica-p2

Contenido

Introducción

En esta memoria se recoge la modelización del problema propuesto en la primera parte como un problema de satisfacción de restricciones (CSP). Además de dicha modelización se incluyen algunas explicaciones sobre la implementación en código utilizando la librería *python-constraint*. A continuación, se realiza un análisis sobre las pruebas propuestas.

Puesto que no iba a poder realizar la segunda parte completa con su respectiva modelización e implementación, he decidido centrarme exclusivamente en la primera parte de la práctica para intentar sacarla de la mejor manera posible.

Por último, se incluye una sección en la que se valora la dificultad y la utilidad de la práctica

Parte 1

Descripción Formal del Modelo

Para modelizar un problema de satisfabilidad de restricciones es necesario definir la terna (*X, D, C*), donde X es el conjunto de variables, D representa los dominios de cada variable, y C es el conjunto de restricciones.

- **Variables:**

El conjunto de variables de decisión dependerá del número de vehículos incluidos en cada test. Es decir, tendremos una variable por cada vehículo que contenga un test concreto.

$$X = \{x_i \; \forall \; i \in vehículosDelTest\}$$

Donde el conjunto *vehículosDelTest* sería por ejemplo en el caso propuesto en el enunciado:

$$vehículosDelTest = \{1\text{-TSU-C}, 2\text{-TNU-X}, 3\text{-TNU-X}, 4\text{-TNU-C}, 5\text{-TSU-X}, 6\text{-TNU-X}, 7\text{-TNU-C}, 8\text{-TSU-C}\}$$

- **Dominios:**

Para definir los dominios de las variables, es conveniente definir previamente dos conjuntos antes.

El conjunto de todas las posibles plazas de parking:

$$plazasParking = \{(x, y) \in \mathbb{N} : 1 \leq x \leq numeroFilas\ ; 1 \leq y \leq numeroColumnas\}$$

Y el conjunto de plazas con electricidad, que es un subconjunto de *plazasParking* y se define para cada test en la segunda línea del fichero.

$$plazasEléctricas \subseteq plazasParking$$

Ahora, es posible definir que:

- El dominio de los vehículos sin congelador es el conjunto de todas las plazas de parking.

  $$D_{vehíuclosSinCongelador} = plazasParking$$

  Donde:

  $$vehículosSinCongelador = \{\text{y-TzU-}\mathbf{X} \mid y \in \mathbb{N};\ z \in \{S, N\}$$

- El dominio de los vehículos con congelador es el conjunto de todas las plazas con electricidad.

  $$D_{vehíuclosConCongelador} = plazasEléctricas$$

  Donde:

  $$vehículosConCongelador = \{\text{y-TzU-}\mathbf{C} \mid y \in \mathbb{N};\ z \in \{S, N\}$$

❖ **<u>Restricciones:</u>**

1. Todo vehículo debe tener asignada sólo una plaza.

   $$x_i \in plazasParking \quad \forall\ i \in vehículosDelTest$$

2. Dos vehículos distintos no pueden ocupar la misma plaza.

   $$x_i \neq x_j \quad \forall\ i, j \in vehículosDelTest$$

3. Los vehículos provistos de congelador sólo pueden ocupar plazas con conexión a la red eléctrica. Esta restricción no es necesaria añadirla explícitamente en la implementación puesto que se cumple por los dominios impuestos.

   $$x_i \in plazasEléctricas \quad \forall\ i \in vehículosConCongelador$$

4. Un vehículo de tipo TSU no puede tener aparcado por delante, en su misma fila, a ningún otro vehículo, excepto si éste también es de tipo TSU.

$$x_i[fila_i][columna_i] \geq x_j[fila_j][columna_j]$$

$$si \quad x_i \in vehículosUrgentes$$

$$y \quad fila_i = fila_j$$

5. Por cuestiones de maniobrabilidad, todo vehículo debe tener libre al menos una plaza a izquierda o derecha (mirando en dirección a la salida).

$$((x_i[fila_i + 1][columna_i] \neq x_j[fila_j][columna_j]) \; or$$

$$(x_i[fila_i - 1][columna_i] \neq x_j[fila_j][columna_j]))$$

$$\forall \, x_i, x_j \in vehículosDelTest$$

Implementación del modelo

Para la implementación del problema en Python se hace uso de la librería *python-constraint*. Dicho recurso permite modelar problemas de satisfabilidad de restricciones y obtener sus soluciones en caso de tenerlas.

Debido a que la librería asigna un único valor (una única plaza) a cada variable (a cada vehículo), no es necesario añadir explícitamente las restricciones 1) y 2) a la implementación.

Además, puesto que la restricción 3) se va a cumplir siempre, al definir los dominios pertinentes a cada tipo de vehículo, esta restricción tampoco es necesario incluirla de forma explícita en la implementación.

En el fichero *.csv* en el que se almacena el número de soluciones, se incluye una solución aleatoria distinta tras cada ejecución del programa.

El fichero que ejecuta todos los tests, contiene en un comentario el comando para ejecutar el test con el caso del enunciado. Se encuentra comentado para no demorar demasiado el tiempo de ejecución del resto de tests.

Análisis de los resultados obtenidos

Para los tests, se ha decidido analizar el caso propuesto en el enunciado, y posteriormente analizar casos con tamaños de parking más limitados que comprueben cada una de las restricciones del problema. Esta reducción, nos permitirá comprobar las diferentes casuísticas del problema sin que el tiempo de ejecución sea demasiado elevado.

- Para el caso del enunciado, se obtiene el siguiente resultado:

Column1	Column2	Column3	Column4	Column5	Column6
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

  Se puede ver que:

  - Todos los vehículos con congelador están en una plaza eléctrica
  - Tos los vehículos tienen al menos un lateral libre
  - Ningún TSU tiene por delante en su misma fila otro vehículo que no sea TSU

En los casos con tamaños más pequeños se va a comprobar que:

- **parking01.txt** → Si el número de vehículos que necesitan congelador excede el número de plazas eléctricas no hay solución.

```
4x4
PE: (1,1) (1,2) (4,1) (4,2)
1-TSU-C
2-TNU-X
3-TNU-X
4-TNU-C
5-TSU-X
6-TNU-C
7-TNU-C
8-TSU-C
```

Column1	Column2	Column3	Column4
N. sol	0		
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-

- **parking02.txt** → Si el número de vehículos supera el número de plazas, no hay solución.

```
3x3
PE: (1,1) (1,2) (2,1) (2,2) (3,1) (3,2)
1-TNU-X
2-TNU-X
3-TNU-X
4-TNU-X
5-TNU-X
6-TNU-X
7-TNU-X
8-TNU-X
9-TNU-X
10-TNU-X
```

Column1	Column2	Column3
N. sol	0	
-	-	-
-	-	-
-	-	-

- **parking03.txt** → Si no hay al menos una plaza libre a cada lado, no hay solución. Para ello forzamos que haya 3 vehículos en la misma columna sin separación.

```
3x3
PE: (1,1) (2,1) (3,1)
1-TSU-C
2-TNU-C
3-TSU-C
```

Column1	Column2	Column3
N. sol	0	
-	-	-
-	-	-
-	-	-

- **parking04.txt** → Si hay un TNU delante de un TSU, no hay solución. Para ello forzamos que tenga que haber un TNU por delante sí o sí de un TSU.

```
2x4
PE: (1,3) (2,4)
1-TNU-C
2-TSU-X
3-TSU-X
4-TNU-C
```

Column1	Column2	Column3	Column4
N. sol	0		
-	-	-	-
-	-	-	-

- **parking05.txt** → Se recibe el número de soluciones esperadas.

```
4x4
PE: (1,1) (4,1)
1-TNU-C
2-TNU-X
4-TNU-C
```

Column1	Column2	Column3	Column4
N. sol	24		
1-TNU-C	-	-	-
-	-	-	-
-	-	-	-
4-TNU-C	2-TNU-X	-	-

En este caso, el vehículo '2-TNU-X' únicamente puede estar en las 12 casillas de las columnas 2, 3 y 4. Pero como los vehículos '1-TNU-C' y '4-TNU-C' pueden permutar entre sí, sabemos que podría haber hasta 24 combinaciones posibles. Lo cual concuerda con la solución dada.

- **parking06.txt** → Si alguna plaza eléctrica no pertenece al conjunto de plazas del parking, salta un error.

```
4x4
PE: (1,1) (6,1)
1-TNU-C
2-TNU-X
4-TNU-C
```

```
Arturo@DESKTOP-TIDE6U8 MINGW64 ~/Desktop/_Heurística y Optimización/Práctica 2/p2-432160/parte-1 (main)
$ py CSPParking.py ./CSP-tests/parking06.txt
Alguna plaza especificada en el fichero de entrada no pertenece al parking del tamaño indicado
```

Estos casos son los más relevantes para comprobar la satisfabilidad de las restricciones propuestas.

Aún así, podría mejorarse el programa haciéndolo resistente a inputs incorrectos tanto en el nombre del parámetro recibido, es decir, en el nombre del archivo, como en los propios datos del archivo. El programa no está preparado para recibir datos incorrectos en un fichero, como que cada fila no se corresponda con el dato esperado o que los datos estén mal especificados.

Sin embargo, estos son casos que el programa no tiene por qué contemplar, pues el enunciado deja claro el formato que deben tener los ficheros de entrada.

Parte 2

Descripción Formal del Modelo

Resultados

Conclusión de la práctica

Me ha parecido una práctica interesante y me habría gustado enfrentarme a la segunda parte. Sin embargo, me ha resultado imposible por cuestiones de tiempo. No porque hubiese poco tiempo para hacer la práctica, sino porque en esta semana se juntan demasiadas entregas y parciales, y si a eso le añades el tener que hacerla en solitario se hace muy cuesta arriba.

Quizás tendría que haber estado más rápido para buscar compañero, o buscar a otro que no dejase la asignatura. Pero en cualquier caso algún compañero se habría quedado solo. Y es realmente complicado hacerla individualmente.

De nuevo, me gustaría recalcar que no entiendo muy bien el sistema de evaluación de la práctica, por el que es imposible rascar puntos en una parte si no está completamente finalizada, tanto en su modelización como en su implementación.