Universidad Carlos III
Heurística y optimización 2022-23
Práctica de programación lineal.
Curso 2022-23

Práctica SAT y CSP.

Date: **15/12/22**

GRUPO REDUCIDO: **83** URL: **https://github.com/palomanr/entrega2_heuristica.git**

Miembros:

Alejandro/González Núñez/100429135

Paloma/Núñez Reyes/100451207

Tárik/Sánchez Ahmed/100451283

TABLA DE CONTENIDOS

1 INTRODUCCIÓN

1.1 Objetivo de la práctica

El objetivo de esta práctica es aprender a modelar problemas de satisfacción de restricciones y de búsqueda heurística.
Se nos dan dos problemas relacionados a modelar, el primero consiste en diseñar la asignación de asientos del alumnado en el autobús. En el segundo caso se plantea conformar la cola de entrada en el autobús de forma que el tiempo que se invierta en dicha operación sea el menor posible.

1.2 Organización del documento

Este documento se divide en las siguientes partes:

- En primera instancia se describen los modelos de los dos problemas mencionados en el objetivo de la práctica. Se separa esta parte en dos subsecciones, primer y segundo modelo.

- En la siguiente sección se ofrecen los análisis de los resultados obtenidos una vez aplicados los modelos descritos anteriormente en el lenguaje de programación de python. Agrupamos el análisis de resultados en dos, que se corresponden con las dos partes del proyecto. Mostramos cómo hemos planteado los modelos de la forma más general posible para ser viables con cualquier tipo de datos introducidos.

- Por último, contamos con una sección dedicada a las conclusiones del proyecto, si consideramos coherentes las soluciones encontradas, comparaciones con respecto a ambos modelos. Además mostramos problemas encontrados y observaciones personales.

2 Descripción modelos

2.1 Primer Modelo

2.1.1 Conjunto de variables X:

Inicialmente definimos las variables directamente de el archivo alumnos.txt que se le indique al problema, donde cada variable tendría el siguiente formato:
“ID,CICLO,CONFLICTIVO,MOV,HERMANO”.

A partir de las variables definimos 10 listas, donde se filtran por (ciclo, movilidad y si son hermanos)

1. Alumnos del ciclo 1 de movilidad normal sin hermano
2. Alumnos del ciclo 2 de movilidad normal sin hermano
3. Alumnos del ciclo 1 de movilidad reducida sin hermano
4. Alumnos del ciclo 2 de movilidad reducida sin hermano
5. Alumnos del ciclo 1 de movilidad normal hermano mismo ciclo

6. Alumnos del ciclo 2 de movilidad normal hermano mismo ciclo
7. Alumnos del ciclo 1 de movilidad reducida hermano mismo ciclo
8. Alumnos del ciclo 2 de movilidad reducida hermano mismo ciclo
9. Alumnos movilidad normal hermano de distinto ciclo
10. Alumnos movilidad reducida hermano de distinto ciclo

Estas variables nos permiten asegurarnos que todos los alumnos se sienten correctamente en los asientos designados y que no se repitan a la hora de agregarlas con addVariables.

2.1.2 Dominios D:

En función de las variables anteriores y de la estructura del bus definimos los dominios para cada una de las variables.

1. Dominio Alumnos del ciclo 1 de movilidad normal sin hermano:

   dom:{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}

2. Dominio Alumnos del ciclo 2 de movilidad normal sin hermano:

   dom{17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32}

3. Dominio Alumnos del ciclo 1 de movilidad reducida sin hermano:

   dom:{1, 2, 3, 4, 13, 14, 15, 16}

4. Dominio Alumnos del ciclo 2 de movilidad reducida sin hermano:

   dom{17, 18, 19, 20}

5. Dominio Alumnos del ciclo 1 de movilidad normal hermano mismo ciclo:

   dom:{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}

6. Dominio Alumnos del ciclo 2 de movilidad normal hermano mismo ciclo:

   dom{17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32}

7. Dominio Alumnos del ciclo 1 de movilidad reducida hermano mismo ciclo:

   dom:{1, 2, 3, 4, 13, 14, 15, 16}

8. Dominio Alumnos del ciclo 2 de movilidad reducida hermano mismo ciclo:

   dom{17, 18, 19, 20}

9. Dominio Alumnos movilidad normal hermano de distinto ciclo:

   Para el mayor: dom:{2, 3, 6, 7, 10, 11, 14, 15}
   Para el menor: dom:{1, 4, 5, 8, 9, 12, 13, 16}

10. Dominio Alumnos movilidad reducida hermano de distinto ciclo:

    dom:{0, 1, 2, 3, 4, 13, 14, 15, 16}

2.1.3 Restricciones R:

Para nuestro planteamiento hemos incorporado 5 restricciones:

1. Asiento al lado de un alumno del ciclo 1 con movilidad reducida debe estar vacío:

Esta función recibe alumnos con movilidad reducida del ciclo 1 sin hermano y prueba que no tenga ningún otro alumno en el asiento que está a la derecha y a la izquierda de él, teniendo en cuenta el pasillo.

2. Asiento al lado de un alumno del ciclo 2 con movilidad reducida debe estar vacío:

Esta función está implementada igual que la anterior sin embargo tiene en cuenta las diferencias entre ambas partes del autobús.

3. Asiento al lado de un alumno mov reducida de ciclo distinto debe estar vacía:

Esta restricción se hace como una forma de solucionar el inconveniente que presenta un alumno de movilidad reducida cuyo hermano no es de movilidad reducida, al no poder sentarse uno al lado del otro.

4. Ningún alumno conflictivo o de movilidad reducida se sienta alrededor de un alumno conflictivo:

Para el funcionamiento correcto de esta restricción primero recibe a todos los alumnos conflictivos y posteriormente comprobar si tiene un hermano, en el caso de que tenga un hermano no le limita sentarse a su hermano a su alrededor, sin embargo a todos los demás alumnos les aplica la restricción.

5. Hermanos juntos y mayor en el pasillo

Como parámetros recibe a los hermanos, comprueba cuál de los 2 es mayor y los envía a la función, la función se encarga de que se sienten juntos y el mayor en el pasillo y si son de ciclos distintos se sienten en la parte delantera

2.2 Segundo Modelo

En este apartado vamos a modelizar la segunda parte de la práctica, donde tenemos que organizar la cola de alumnos para entrar en el autobús de la manera más eficiente posible. Para hacer esto tenemos que tener en cuenta lo siguiente:

- Los alumnos de movilidad reducida tardan tres veces más en montar el autobús que el resto de alumnos.
- No puede haber un alumno de movilidad reducida detrás de otro de movilidad reducido
- No puedo ver un alumno de movilidad reducida al final de la cola
- El alumno detrás del alumno con movilidad reducida tarda lo mismo que el en montar el autobús dado que le tiene que ayudar
- Un alumno conflictivo duplica el tiempo necesario para subir al autobús al alumno de delante como al alumno de detrás
- Un alumno conflictivo duplica el tiempo necesario a todos los alumnos detrás de él con un número mayor de asiento asignado

Hemos descrito los siguientes elementos:

- Ai: Alumnos en estado inicial
- Af: Alumnos en estado final

1. Dos estados, inicial y final que se corresponden con una terna: (Ai, Af) en la que: Ai, Af $\in$ [0, . . . , 32] indican el número de alumnos que quedan por organizar en la cola del autobús, como mucho podemos contar con 32 alumnos.
2. El estado inicial se representa como la terna donde inicialmente tenemos la cola ([lista id alumnos colocada de menor a mayor según el asiento], i, [0]) y el final como ([0],f,[lista id alumnos colocada de la forma más eficiente])
3. Se dispone de un operador: Mover1A(x,y).

Inicialmente íbamos a poner un operador por cada alumno, pero nos percatamos de que era mejor tener uno único, *mover un alumno* sea el que sea y ya posteriormente hacer una función que se encargue de diferenciar si el alumno que se está moviendo resulta conflictivo, de movilidad reducida o ninguno de los anteriores en ese único operador.

Un ejemplo de cómo podría ser dado un archivo como: {'1XR': 1, '8CR': 19, '3XX': 21, '6XX': 22, '4CX': 32}

- Estado inicial: (INICIAL: {'1XR': 1, '8CR': 19, '3XX': 21, '6XX': 22, '4CX': 32}, i, [0])
- Estado final: ([0], f, {'1XR': 1, '4CX': 32, '3XX': 21, '8CR': 19, '6XX': 22})

El modelo implementado en nuestro código está de la siguiente manera:

- Contamos con 4 funciones, una se corresponde a la lectura de la entrada introducida y las siguientes tres a la escritura de lols archivos de salida, uno correspondiente con las estadísticas y otro con el inicial *input* y el *output* final.
- Tenemos tres clases, la primera que se vería al abrir el archivo *ASTARColaBus.py* es la correspondiente a los alumnos donde se obtienen los identificadores y las características, si es conflictivo o no y si es de movilidad reducida o no.

- En la clase *Estado* hace referencia a los estados que s evan consiguen al iniciar el algoritmo y progresivamente hasta que se finaliza. Tenemos una función para calcular *g* con las restricciones que se nos describían en el enunciado referidas a los costes a tener en cuenta. Después contamos con 4 heurísticas que hemos podido derivar y luego con funciones adicionales como puede ser la de expandir nodos.
- La última clase es la correspondiente al algoritmo A*.

3 Análisis de los resultados

En este apartado vamos a analizar los resultados de las pruebas que hemos realizado para las dos partes.

3.1 Primera Parte

- **Prueba 1 (Parte delantera del bus llena):**

  El propósito de esta prueba es llenar la parte delantera del bus con todo los distintos tipos de alumnos posibles combinando los campos de: movilidad, conflictivo y hermanos (Tanto de ciclos distintos como del mismo ciclo)

Alumnos introducidos	1,1,X,R,0 2,1,X,R,0 3,1,C,R,0 4,2,X,X,7 5,2,X,X,8 6,2,X,X,9 7,1,X,X,4 8,1,X,X,5 9,1,X,X,6 10,1,C,R,11 11,1,C,X,10 12,1,X,X,0
Total de alumnos	12
Número de soluciones	768
Una solución:	{'1XR': 2, '10CR': 4, '8XX': 5, '5XX': 6, '12XX': 7, '11CX': 8, '9XX': 9, '6XX': 10, '4XX': 11, '7XX': 12, '2XR': 14, '3CR': 16}

- **Prueba 2 (Parte trasera del bus llena):**
  Esta prueba es esencialmente igual a la anterior sin embargo al estar distribuida de diferente manera se nos podemos asegurar que cumpla con la distribución del bus.

Alumnos introducidos	1,2,C,R,2 2,2,C,R,1 3,2,C,X,4 4,2,C,X,3 5,2,C,X,0 6,2,X,X,7 7,2,X,X,6 8,2,X,X,9 9,2,X,X,8 10,2,X,X,11 11,2,X,X,10 12,2,X,X,0 13,2,X,X,14 14,2,X,X,13
Total de alumnos	14
Número de soluciones	6144
Una solución:	{'2CR': 18, '1CR': 19, '6XX': 21, '7XX': 22, '8XX': 23, '9XX': 24, '3CX': 25, '4CX': 26, '10XX': 27, '11XX': 28, '13XX': 29, '14XX': 30, '12XX': 31, '5CX': 32}

- **Prueba 3 (Parte delantera y trasera del bus medio llena):**
  Para esta prueba queríamos llenar ambas partes, sin embargo al generarse demasiadas soluciones posibles, el programa la librería de python constraints se demora demasiado. Si se imprime una única solución con el autobús completamente lleno lo hace sin problemas. Es por todo esto que esta prueba mezcla un poco de las anteriores 2 pero sin demasiados alumnos

Alumnos introducidos	1,1,X,R,0 2,1,X,R,0 3,1,C,R,0 4,2,X,X,7 5,2,X,X,8 6,2,X,X,9 7,1,X,X,4 8,1,X,X,5 9,1,X,X,6 10,1,X,R,11 11,1,X,X,10 12,2,X,R,13 13,2,X,R,12 14,2,C,X,15 15,2,C,X,14 16,2,C,X,0
Total de alumnos	16
Número de soluciones	73 728
Una solución:	{'2XR': 2, '3CR': 4, '8XX': 5, '5XX': 6, '11XX': 7, '9XX': 9, '6XX': 10, '4XX': 11, '7XX': 12, '10XR': 14, '1XR': 16, '12XR': 18, '13XR': 19, '14CX': 25, '15CX': 26, '16CX': 32}

- **Prueba 4 (Solo alumnos de movilidad normal):**
  Con este test queremos comprobar que los alumnos de movilidad normal también ocupan asientos de movilidad reducida tanto en la parte delantera como trasera del bus, incluso siendo conflictivos si el caso lo permite.

Alumnos introducidos	1,1,X,X,0 2,1,X,X,0 3,1,C,X,0 4,2,X,X,7 5,2,X,X,8 6,2,X,X,9 7,1,X,X,4 8,1,X,X,5 9,1,X,X,6 10,2,C,X,11 11,2,C,X,10
Total de alumnos	11
Número de soluciones	3 598 560
Una solución:	{'7XX': 1, '4XX': 2, '6XX': 3, '9XX': 4, '2XX': 5, '1XX': 13, '3CX': 14, '5XX': 15, '8XX': 16, '11CX': 31, '10CX': 32}

- **Prueba 5 (Solo alumnos conflictivos):**
  Con este test queremos comprobar que los alumnos conflictivos se sientan alejados de los demás conflictivos o de movilidad reducida, así también comprobar que si los 2 conflictivos son hermanos estos se sentaran uno al lado del otro sean de ciclos distintos o del mismo.

Alumnos introducidos	1,1,C,R,0 2,1,C,R,0 3,1,C,X,0 4,2,C,X,7 5,2,C,X,8 6,2,C,X,0 7,1,C,X,4 8,2,C,X,5 9,2,C,X,0
Total de alumnos	9
Número de soluciones	264
Una solución:	{'2CR': 1, '3CX': 3, '7CX': 9, '4CX': 10, '1CR': 16, '6CX': 23, '9CX': 25, '8CX': 31, '5CX': 32}

3.2 Segunda Parte

- Heuristica 1: Sumar 1 por cada alumno sin estar en la cola.

Pruebas	Alumnos1	Alumnos2
Cola Inicial	{'1XX': 5, '8XR': 13, '3CX': 22, '6XX': 23, '4CX': 32}	{'1XR': 1, '8CR': 15, '3CR': 17, '6XX': 22, '4XX': 27, '5XX': 31}
Cola Final	{'8XR': 13, '1XX': 5, '6XX': 23, '4CX': 32, '3CX': 22}	{'3CR': 17, '6XX': 22, '1XR': 1, '4XX': 27, '8CR': 15, '5XX': 31}
Tiempo Total	0.002992	0.001997
Coste Total	9	48
Longitud del plan	5	6
Nodos expandidos	132	85

Pruebas	Alumnos3	Alumnos4
Cola Inicial	{'1XR': 1, '8CR': 19, '3XX': 21, '6XX': 22, '4CX': 32}	{'4XR': 14, '8CR': 20, '2XX': 28, '5XX': 29, '6XX': 31, '1CX': 32}
Cola Final	{'1XR': 1, '4CX': 32, '3XX': 21, '8CR': 19, '6XX': 22}	{'1CX': 32, '2XX': 28, '4XR': 14, '5XX': 29, '8CR': 20, '6XX': 31}
Tiempo Total	0.001998	0.015997
Coste Total	22	21
Longitud del plan	5	6
Nodos expandidos	65	393

Pruebas	Alumnos5

Cola Inicial	{'4CR': 1, '1CX': 11, '3XX': 12, '2XX': 21, '5XX': 22, '6XR': 23}
Cola Final	{'6XR': 23, '3XX': 12, '4CR': 1, '2XX': 21, '5XX': 22, '1CX': 11}
Tiempo Total	0.014001
Coste Total	24
Longitud del plan	6
Nodos expandidos	369

- Heuristica 2: Quitar que los de movilidad reducida se les multiplique por dos

Pruebas	Alumnos1	Alumnos2
Cola Inicial	{'1XX': 5, '8XR': 13, '3CX': 22, '6XX': 23, '4CX': 32}	{'1XR': 1, '8CR': 15, '3CR': 17, '6XX': 22, '4XX': 27, '5XX': 31}
Cola Final	{'8XR': 13, '1XX': 5, '6XX': 23, '4CX': 32, '3CX': 22}	{'3CR': 17, '6XX': 22, '1XR': 1, '4XX': 27, '8CR': 15, '5XX': 31}
Tiempo Total	0.002999	0.001995
Coste Total	9	48
Longitud del plan	5	6
Nodos expandidos	132	85

Pruebas	Alumnos3	Alumnos4
Cola Inicial	{'1XR': 1, '8CR': 19, '3XX': 21, '6XX': 22, '4CX': 32}	{'4XR': 14, '8CR': 20, '2XX': 28, '5XX': 29, '6XX': 31, '1CX': 32}

Cola Final	{'1XR': 1, '4CX': 32, '3XX': 21, '8CR': 19, '6XX': 22}	{'1CX': 32, '2XX': 28, '4XR': 14, '5XX': 29, '8CR': 20, '6XX': 31}
Tiempo Total	0.001001	0.011998
Coste Total	22	21
Longitud del plan	5	6
Nodos expandidos	59	384

Pruebas	Alumnos5
Cola Inicial	{'4CR': 1, '1CX': 11, '3XX': 12, '2XX': 21, '5XX': 22, '6XR': 23}
Cola Final	{'6XR': 23, '3XX': 12, '4CR': 1, '2XX': 21, '5XX': 22, '1CX': 11}
Tiempo Total	0.013564
Coste Total	24
Longitud del plan	6
Nodos expandidos	363

- Heuristica 3: Quitar que los coflictivos se multipliquen por tres.

Pruebas	Alumnos1	Alumnos2
Cola Inicial	{'1XX': 5, '8XR': 13, '3CX': 22, '6XX': 23, '4CX': 32}	{'1XR': 1, '8CR': 15, '3CR': 17, '6XX': 22, '4XX': 27, '5XX': 31}
Cola Final	{'8XR': 13, '1XX': 5,	{'3CR': 17, '6XX': 22,

	'6XX': 23, '4CX': 32, '3CX': 22}	'1XR': 1, '4XX': 27, '8CR': 15, '5XX': 31}
Tiempo Total	0.002998	0.002001
Coste Total	9	48
Longitud del plan	5	6
Nodos expandidos	130	85

Pruebas	Alumnos3	Alumnos4
Cola Inicial	{'1XR': 1, '8CR': 19, '3XX': 21, '6XX': 22, '4CX': 32}	{'4XR': 14, '8CR': 20, '2XX': 28, '5XX': 29, '6XX': 31, '1CX': 32}
Cola Final	{'1XR': 1, '4CX': 32, '3XX': 21, '8CR': 19, '6XX': 22}	{'1CX': 32, '2XX': 28, '4XR': 14, '5XX': 29, '8CR': 20, '6XX': 31}
Tiempo Total	0.000994	0.014999
Coste Total	22	21
Longitud del plan	5	6
Nodos expandidos	65	393

Pruebas	Alumnos5
Cola Inicial	{'4CR': 1, '1CX': 11, '3XX': 12, '2XX': 21, '5XX': 22, '6XR': 23}
Cola Final	{'6XR': 23, '3XX': 12, '4CR': 1, '2XX': 21, '5XX': 22, '1CX': 11}
Tiempo Total	0.014
Coste Total	24

Longitud del plan	6
Nodos expandidos	363

- Heuristica 4: Quitar que tanto a conflictivos como a los de movilidad reducida se les multiplique por 3 y por 2 respectivamente

Pruebas	Alumnos1	Alumnos2
Cola Inicial	{'1XX': 5, '8XR': 13, '3CX': 22, '6XX': 23, '4CX': 32}	{'1XR': 1, '8CR': 15, '3CR': 17, '6XX': 22, '4XX': 27, '5XX': 31}
Cola Final	{'8XR': 13, '1XX': 5, '6XX': 23, '4CX': 32, '3CX': 22}	{'3CR': 17, '6XX': 22, '1XR': 1, '4XX': 27, '8CR': 15, '5XX': 31}
Tiempo Total	0.002996	0.001998
Coste Total	9	48
Longitud del plan	5	6
Nodos expandidos	130	85

Pruebas	Alumnos3	Alumnos4
Cola Inicial	{'1XR': 1, '8CR': 19, '3XX': 21, '6XX': 22, '4CX': 32}	{'4XR': 14, '8CR': 20, '2XX': 28, '5XX': 29, '6XX': 31, '1CX': 32}
Cola Final	{'1XR': 1, '4CX': 32, '3XX': 21, '8CR': 19, '6XX': 22}	{'1CX': 32, '2XX': 28, '4XR': 14, '5XX': 29, '8CR': 20, '6XX': 31}
Tiempo Total	0.000997	0.01799
Coste Total	22	21
Longitud del plan	5	6
Nodos expandidos	59	384

Pruebas	Alumnos5
Cola Inicial	{'4CR': 1, '1CX': 11, '3XX': 12, '2XX': 21, '5XX': 22, '6XR': 23}
Cola Final	{'6XR': 23, '3XX': 12, '4CR': 1, '2XX': 21, '5XX': 22, '1CX': 11}
Tiempo Total	0.013997
Coste Total	24
Longitud del plan	6
Nodos expandidos	357

Con todas estas pruebas, vemos como la mejor heurística se corresponde con la última, la cuarta, donde suprimimos que tanto a conflictivos como a los de movilidad reducida se les multiplique por 3 y por 2 respectivamente. En este caso se llega a la solución expandiendo menor número de nodos y por lo tanto tiene un tiempo de ejecución ligeramente menor con respecto al resto. Esta heurística es la mejor ya que relajamos un número adecuado de restricciones y conseguimos una misma solución adecuada pero de una forma más eficiente.

4 Conclusiones

Como conclusiones finales podemos decir que con respecto a la primera parte, ha sido un trabajo difícil el poder asignar a todos los alumnos a un asiento cumpliendo todas las restricciones, sobretodo la restricción de que que sean hermanos ya que esta limita muchísimo el problemas modificando las restricciones anteriores y creando nuevas restricciones que fácilmente pueden dar error si no se estudian al detalle. La librería de python constraints funciona muy bien sin embargo sería bueno una opción intermedia entre obtener una sola solución o obtener todas las soluciones, ya que algunas de las pruebas no se han podido mostrar debido a que existen millones de posibles soluciones al problema. Aun así ha sido muy satisfactorio el resolver el problema y ver que todo se ejecuta correctamente.
En cuanto a la segunda parte del proyecto, definir el modelo fue relativamente sencillo, la implementación en el código no fue igual ya que consideramos que A* no

es la mejor opción para este problema, ya que lo consideramos un problema de optimización más que de búsqueda. Después de hacer las pruebas, vemos como la cuarta heurística, donde relajamos la restricciones que triplican y duplican los costes de los alumnos conflictivos y de movilidad reducida, resulta ser la mejor al expandir menos nodos. Hemos conseguido una solución adecuada pero de una forma más eficiente con esta heurística.