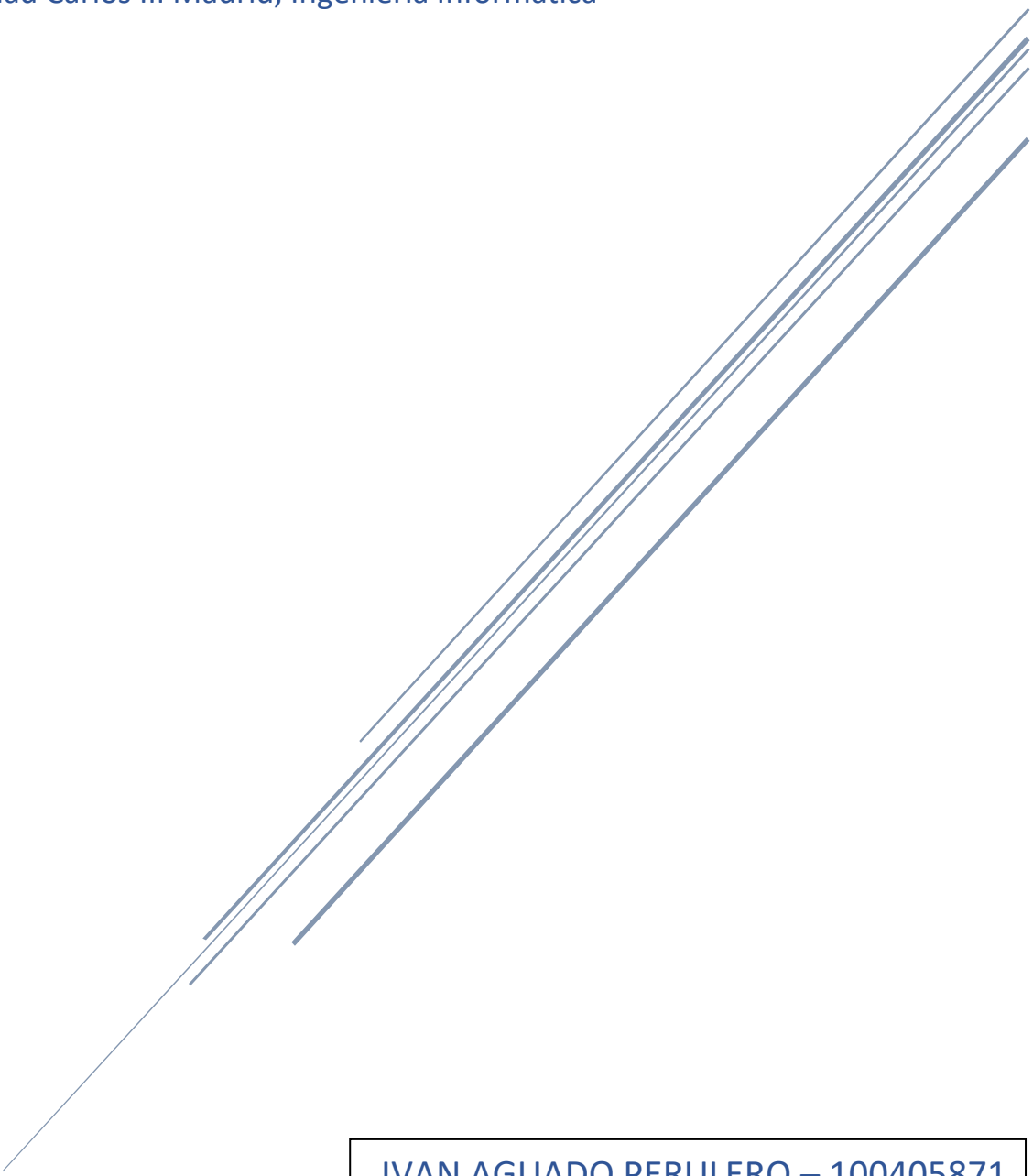
HEURÍSTICA Y OPRIMIZACIÓN

PRÁCTICA 2

Universidad Carlos III Madrid, Ingeniería Informática



IVAN AGUADO PERULERO – 100405871
JORGE SERRANO PÉREZ – 100405987

Contenido

1.- INTRODUCCIÓN

El objetivo de esta práctica es modelar y resolver problemas tanto de satisfacción de restricciones como de búsqueda heurística.

En primer lugar, se realizará una descripción de los modelos que se han desarrollado de acuerdo con las restricciones expuestas en el enunciado. Se añadirán las decisiones de diseño que hemos considerado nosotros para resolver eficientemente ambos problemas.

En segundo lugar, se procederá con la implementación de esos modelos en el lenguaje de programación *Python* junto con la librería *python-constraint*.

Por otro lado, se realizarán una serie de casos de prueba con el fin de analizar el comportamiento del programa y observar cuáles son las restricciones más importantes. Además, en la segunda parte, se realizará también un estudio comparativo utilizando dos heurísticas diferentes.

Por último, se añadirán distintas conclusiones personales que hemos obtenido durante la realización de la práctica.

2.- PARTE 1: CSP

MODELADO

En este primer problema podemos distinguir la siguiente información obtenida del enunciado:

- Existen 6 satélites distintos que disponen de una o varias franjas horarias en las que son "visibles" por una o varias antenas de transmisión.
- Las franjas horarias son muy variadas: de 00:00 a 12:00, de 13:00 a 16:00, etc.
- Existen 12 antenas distintas

Cada satélite es visible por las siguientes antenas:

Satélite	Franja	Antenas
SAT1	00:00 - 12:00	ANT1, ANT2, ANT3, ANT4
SAT2	00:00 - 12:00	ANT1, ANT2, ANT3
SAT3	06:00 - 12:00	ANT4, ANT6
SAT3	13:00 - 16:00	ANT7, ANT9, ANT10
SAT4	16:00 - 00:00	ANT8, ANT11, ANT12
SAT5	06:00 - 13:00	ANT1, ANT7, ANT12
SAT6	09:00 - 13:00	ANT7, ANT9
SAT6	13:00 - 19:00	ANT3, ANT4, ANT5

Se nos pide determinar cuál es la asignación de satélites y sus franjas horarias a antenas de transmisión. Por ello, hemos decidido que las variables de decisión son las siguientes:

- SAT1_F1, SAT2_F1, SAT3_F1, SAT3_F2, SAT4_F1, SAT5_F1, SAT6_F1, SAT6_F2.

Cada una se refiere a un satélite "SAT" en una determinada franja horaria "F". Los satélites 5 y 6 tienen dos horarios posibles por eso existen dos variables para cada uno, uno que representa al primer intervalo de tiempo posible "F1" y otro para el segundo "F2".

El dominio de cada una de las variables se expondrá a continuación, a cada satélite en una determinada franja horaria se le asignan las antenas que tiene visibles. Las antenas "ANT" van desde la número 1 hasta la 12.

- D(SAT1_F1) = {ANT1, ANT2, ANT3, ANT4}
- D(SAT2_F1) = {ANT1, ANT2, ANT3}
- D(SAT3_F1) = {ANT4, ANT6}
- D(SAT3_F2) = {ANT7, ANT9, ANT10}
- D(SAT4_F1) = {ANT8, ANT11, ANT12}
- D(SAT5_F1) = {ANT1, ANT7, ANT12}
- D(SAT6_F1) = {ANT7, ANT9}
- D(SAT6_F2) = {ANT3, ANT4, ANT5}

Por otro lado, las restricciones propuestas en el enunciado nos han resultado de esta manera:

1. Todos los satélites deben tener asignada una antena de transmisión en tierra para cada una de sus franjas horarias visibles.

   Se tiene en cuenta al existir una variable de decisión diferente para cada franja horaria de un mismo satélite.

2. A los satélites SAT1 y SAT2 se les debe asignar la misma antena de transmisión.

   SAT1_F1 = SAT2_F1

3. Los satélites SAT2, SAT4 y SAT5 deben tener asignados antenas de transmisión diferentes.

   SAT2_F1 ≠ SAT4_F1 ≠ SAT5_F1

4. Si SAT5 se comunica con ANT12, SAT4 no se puede comunicar con ANT11.

   En este caso, establecemos la restricción como los pares de valores válidos:

   $R_{SAT5\_F1,SAT4\_F1}$ = {(ANT1, ANT8), (ANT1, ANT11), (ANT1, ANT12)

   (ANT7, ANT8), (ANT7, ANT11), (ANT7, ANT12)

   (ANT12, ANT8), (ANT12, ANT12)}

5. Si en una solución se asignan las antenas ANT7 y ANT12, se deben asignar ambas franjas horarias que comiencen antes de las 12:00 o franjas horarias que comiencen después de las 12:00.

Al igual que la anterior, establecemos la restricción como los pares de valores válidos, para cada par de satélites involucrados:

$R_{SAT3\_F2,SAT5\_F1}$ = {(ANT7, ANT1), (ANT7, ANT7),

(ANT9, ANT1), (ANT9, ANT7), (ANT9, ANT12)

(ANT10, ANT1), (ANT10, ANT7), (ANT10, ANT12)}

$R_{SAT4\_F1,SAT5\_F1}$ = {(ANT8, ANT1), (ANT8, ANT7), (ANT8, ANT12),

(ANT11, ANT1), (ANT11, ANT7), (ANT11, ANT12),

(ANT12, ANT1), (ANT12, ANT12)}

$R_{SAT4\_F1,SAT6\_F1}$ = {(ANT8, ANT7), (ANT8, ANT9),

(ANT11, ANT7), (AN11, ANT9),

(ANT12, ANT9}

En cuanto a las restricciones para el orden opuesto de los satélites, por ejemplo, la restricción $R_{SAT5\_F1,SAT3\_F2}$ sería igual que la definida $R_{SAT3\_F2,SAT5\_F1}$ pero cambiando el orden de las antenas en los pares de valores válidos. De igual manera pasaría en el resto.

IMPLEMENTACIÓN

En cuanto a la implementación, hemos decidido en primer lugar crear 8 variables de decisión, que se refieren a los satélites y las franjas horarias. Por eso son 8, ya que hay dos satélites que tienen 2 franjas horarias. Además, los hemos juntado todos en un array para agrupar las variables y hacer el modelo más general, en el que solo tengas que cambiar la variable en un lugar en caso de que esa variable se use en más de una restricción. Hemos hecho lo mismo con las 12 antenas disponibles.

Por otro lado, queremos destacar que hemos hecho uso de una funcionalidad que tienen los *arrays* en *python*, que es la de seleccionar un rango de valores en vez de tener que ir uno a uno (Ej: array [:2] = *array*[0] y *array*[1]). Sin embargo, solo lo hemos podido utilizar con aquellos valores que son consecutivos en el *array*. Por ejemplo, para definir el dominio del primer satélite, según la tabla las antenas disponibles son las 4 primeras, por lo que hemos podido utilizar esta funcionalidad. Pero para el satélite 4, como los valores no son consecutivos hemos tenido que insertar el dominio uno a uno.

Para las 3 primeras restricciones hemos hecho uso de funciones de *python-constraint*. La primera hace referencia al dominio de las variables. La segunda emplea la función *AllEqualConstraint()* para asegurarse de que tienen la misma antena, mientras que las tercera usa *AllDifferentConstraint()* para lo opuesto.

Para la cuarta restricción, hemos creado una función que compruebe que, si las antenas seleccionadas son la 12 y la 11, que devuelva el valor *False*.

Por último, para la última restricción, hemos decidido dividir los satélites en aquellos que tienen la franja horaria asignada antes de las 12:00 o después. Lo siguiente es crear una restricción para comprobar si las antenas son la 7 y la 12 o viceversa, que devolverá *False*.

CASOS DE PRUEBA

Para analizar el comportamiento del programa, vamos a realizar distintos casos de prueba:

- En primer lugar, hemos probado a eliminar un satélite, lo que nos muestra que el número de soluciones baja en gran cantidad. Esto se debe a que existen menos combinaciones posibles.
- Lo mismo ocurre si eliminamos una antena, el número de soluciones se reduce, aunque no tanto como en el caso anterior.
- El caso opuesto es añadir un satélite, que aumenta exponencialmente el número de soluciones.
- Lo mismo sucede de nuevo si añadimos una antena, el número de soluciones aumenta, pero no tanto.
- Si eliminamos distintas restricciones, vemos que la más importante es la primera, ya que aumentan mucho los resultados posibles. Además, observamos también que la segunda es la menos importante, ya que, si la eliminamos, el resultado no varía en exceso.
- Si forzamos que el dominio de cualquier variable este vacío, forzamos la primera restricción, en la que en cada una de las franjas del satélite se le tiene que asignar una antena de transmisión.
- Si forzamos a que el dominio del satélite 1 sea de una sola antena, vemos que se reduce el número de soluciones, por lo que observamos que la restricción 2 se cumple.
- Si forzamos a que el dominio de los satélites 2 y 4 sea el mismo, esto hace que el modelo no encuentre ninguna solución, por lo que la restricción número 3 funciona correctamente.
- Si forzamos a que el dominio del satélite 5 sea la antena 12 y el dominio del satélite 4 sea la antena 11, el modelo no encuentra solución, lo que nos indica que la restricción 4 se cumple.
- Si forzamos que el satélite 3 en la franja 1 tenga en su dominio solo la antena 7, y que el satélite 4 tenga en su dominio la antena 12, el problema es insatisfacible y el modelo no encuentra solución.

3.- PARTE 2: BÚSQUEDA HEURÍSTICA

MODELADO

En este segundo problema podemos distinguir la siguiente información obtenida del enunciado:

- Cada satélite tiene asociada una banda de observación formada por dos filas consecutivas.
- Los satélites solo están visibles 12h al día.
- Las acciones que puede realizar un satélite son: no hacer nada, tomar observaciones, transmitir a la base, girar y cargar energía. Cada una de estas acciones requiere de 1h para su realización.
- Mediante el giro el satélite SAT1 puede cambiar su banda de observación de las filas (0,1) a la (1,2) y viceversa, y el SAT2 puede cambiar su banda de observación de las filas (2,3) a la (1,2) y viceversa.
- Todas las operaciones tienen un coste de una unidad de energía y la acción de carga recupera una unidad.
- El SAT1 tiene una batería de 1 unidad y SAT2 de 8 unidades.
- En casos de observaciones consecutivas, si el satélite no cuenta con la carga suficiente, deberá usar ese turno para recargar. Por lo tanto, puede darse el caso de que los satélites transiten por las 12h de sombra, antes de volver a reanudar con sus operaciones.

Tras extraer los datos y analizar el problema, el modelado del mismo es el siguiente:

Estados:

E = (SAT1, SAT2, objLeft)

- *SAT1 = (colHora, batDisponible, batTotal, objObserved, fila1, fila2)*

Indica dónde se encuentra el satélite 1 mediante:

- *colHora* ∈ [0,1, 2, ...,11] columna en la que se encuentra el satélite 1, cada una representa una hora.
- *batDisponible* ∈ [0,1] representa la batería que le queda al satélite 1 en un momento determinado.
- *batTotal* ∈[1,8] representa la batería máxima del satélite 1
- *objObserved* = (O_1, O_2, O_3...) lista de objetivos observados por el satélite 1.

Como una banda de observación de un satélite está compuesta por dos filas diferentes:

- *fila1* ∈ [0,1] fila superior perteneciente a la banda de observación del satélite 1.
- *fila2* ∈ [1,2] fila inferior perteneciente a la banda de observación del satélite 1.

- *SAT2 = (colHora, batDisponible, batTotal, objObserved, fila1, fila2)*

Indica dónde se encuentra el satélite 2 mediante:

- *colHora* ∈ [0,1, 2, ...,11] columna en la que se encuentra el satélite 2, cada una representa una hora.
- *batDisponible* ∈ [0,1, 2, ...,8] representa la batería que le queda al satélite 2 en un momento determinado.
- *batTotal* ∈[1,8] representa la batería máxima del satélite 2

- *objObserved* = (O_1, O_2, O_3...) lista de objetivos observados por el satélite 2

Como una banda de observación de un satélite está compuesta por dos filas diferentes:

- *fila1* ∈ [1,2] fila superior perteneciente a la banda de observación del satélite 2.
- *fila2* ∈ [2,3] fila inferior perteneciente a la banda de observación del satélite 2.

- *objLeft* = (O_1, O_2, O_3...) lista de objetivos que quedan por observar.

Cada uno de los objetivos a su vez cuenta con:

O_i*= (colHoraObj, filaObj)*

- *colHoraObj* ∈ [0,1, 2, ...,11] columna en la que se encuentra el objetivo i, cada una representa una hora.
- *filaObj* ∈ [0, ...,3] fila en la que se encuentra el objetivo i.

De esta manera el estado inicial para el problema ejemplo propuesto sería:

E_0*= (SAT1, SAT2, objLeft)*

Donde:

SAT1 = (0, 1, 1, (), 0, 1)

SAT2 = (0, 8, 8, (), 2, 3)

objLeft = (O_1, O_2, O_3)

A su vez, el estado final sería:

E_f*= (SAT1, SAT2, objLeft)*

Donde:

SAT1 = (7, 0, 1, (), 0, 1)

SAT2 = (7, 5, 8, (), 1, 2)

objLeft = ()

Acciones:

- *Idle*: Nada
- *Observe*: Tomar observaciones del terreno
- *Transfer*: Transmitir observaciones a la estación base
- *Turn*: Girar para cambiar su banda de observación
- *Charge*: Realizar operaciones de carga de energía

Operadores	Precondiciones	Efectos	Coste
Observe(x, y)	$colHora_x < 12 \wedge batDisponible_x >= 1 \wedge ((fila1_x = filaObj_y \wedge colHora_x = colHoraObj_y) \vee$	$colHora_x = (colHora_x + 1) \% 24 \wedge batDisponible_x = batDisponible_x - 1 \wedge objLeft =$	1 unidad de batería.

	$(fila2_x = filaObj_y \wedge colHora_x = colHoraObj_y) \wedge objLeft \neq ()$	objLeft - y	
Transfer(x,y)	$colHora_x < 12 \wedge batDisponible_x >= 1 \wedge objObserved_x \neq ()$	$colHora_x = (colHora_x + 1) \% 24 \wedge batDisponible_x = batDisponible_x - 1 \wedge objObserved_x = objObserved_x - y$	1 unidad de batería.
TurnDownSatSuperior(x)	$colHora_x < 12 \wedge batDisponible_x >= 1 \wedge fila1_x = 0 \wedge fila2_x = 1$	$fila1_x = fila1_x + 1 \wedge fila2_x = fila2_x + 1 \wedge colHora_x = (colHora_x + 1) \% 24 \wedge batDisponible_x = batDisponible_x - 1$	1 unidad de batería.
TurnDownSatInferior(x)	$colHora_x < 12 \wedge batDisponible_x >= 1 \wedge fila1_x = 1 \wedge fila2_x = 2$	$fila1_x = fila1_x + 1 \wedge fila2_x = fila2_x + 1 \wedge colHora_x = (colHora_x + 1) \% 24 \wedge batDisponible_x = batDisponible_x - 1$	1 unidad de batería.
TurnUpSatSuperior(x)	$colHora_x < 12 \wedge batDisponible_x >= 1 \wedge fila1_x = 1 \wedge fila2_x = 2$	$fila1 = fila1 - 1 \wedge fila2 = fila2 - 1 \wedge colHora_x = (colHora_x + 1) \% 24 \wedge batDisponible_x = batDisponible_x - 1$	1 unidad de batería.
TurnUpSatInferior(x)	$colHora_x < 12 \wedge batDisponible_x >= 1 \wedge fila2_x = 1 \wedge fila2_x = 3$	$fila1_x = fila1_x - 1 \wedge fila2_x = fila2_x - 1 \wedge colHora_x = (colHora_x + 1) \% 24 \wedge batDisponible_x = batDisponible_x - 1$	1 unidad de batería.
Charge(x)	$colHora_x < 12 \wedge batDisponible_x < batTotal_x$	$batDisponible_x = batDisponible_x + 1 \wedge colHora_x = (colHora_x + 1) \% 24$	0 unidades de batería.

Heurísticas:

- Simple

Relajamos la precondición de la batería disponible, es decir, asumimos que un satélite tiene batería infinita. A su vez suponemos que dos objetivos no pueden encontrarse en filas consecutivas, dentro de una misma columna, por lo que un satélite no necesitaría girar, ya que el SAT1 cubriría las filas 1 y 2 y el SAT2 las 3 y 4.

h(n) = m + 1

Donde:

- m = max(colHoraObj[])
- colHoraObj[] es una lista con todas las posiciones en cuanto a columna de los objetivos.

A la columna más alejada en la que se encuentra un objetivo se le suma 1 para tener en cuenta el turno que se emplea en transmitir esa observación.

- Avanzada

Esta heurística tampoco tiene en cuenta la batería disponible, pero si la posición del satélite en un momento determinado. De esta manera podremos sumar 1 cuando se vaya a observar un objetivo y esté fuera de la franja de observación de ambos satélites, es decir, se tienen en cuenta los giros.

Matemáticamente resultaría de la siguiente manera:

$$\mathbf{h(n) = m + 1 + (\sum_{i=0}^{m} min(|fila1_i - filaObjSuperior_i|, |fila2_i - filaObjSuperior_i|) * hayObjetivoSuperior[i] + min(|fila1_i - filaObjInferior_i|, |fila2_i - filaObjInferior_i|) * hayObjetivoInferior[i])}$$

Dónde:

- *fila1* y *fila2* del primer mínimo se refieren al SAT1 y los del segundo mínimo al SAT2.
- *filaObjSuperior* = 0 *filaObjInferior* = 3. Esto es debido a que son las únicas posiciones que pueden quedar fuera del alcance de ambos satélites en un turno determinado.
- *hayObjetivoSuperior*[i], lista de 1´s y 0´s. Un 1 indica que hay un objetivo en la columna i, y un 0 que no, para la fila superior o 0.
- *hayObjetivoInferior*[i], lista de 1´s y 0´s. Un 1 indica que hay un objetivo en la columna i, y un 0 que no, para la fila inferior o 3.

IMPLEMENTACIÓN

En la implementación de esta parte, simplemente se ha seguido el modelo descrito anteriormente. Se han creado los estados inicial y final, junto a cada satélite con sus respectivas variables.

Queremos destacar el diseño que hemos hecho de los distintos métodos que comprueban las precondiciones (*preconditionsTurn()*, ...), que es lo primero que se comprueba antes de realizar los efectos descritos en la tabla anterior.

A continuación, se ha implementado el algoritmo A* de búsqueda heurística con las que hemos descrito en el punto previo a este.

4.- CONCLUSIONES

En la primera parte, hemos aprendido a utilizar la biblioteca python-constraint, que nos ha ayudado a modelizar más fácilmente los problemas de satisfabilidad de restricciones de manera rápida y sencilla. Además, nos hemos dado cuenta de la importancia de modelizar correctamente los problemas propuestos, ya que una vez has dado ese paso el resto es muy simple.

En cuanto a la segunda, hemos decidido utilizar Python 3 para resolver el problema, ya que es un lenguaje muy simple de leer al no ser estrictamente tipado, pero que además incluye listas dinámicas para el algoritmo A* y programación orientada a objetos. Nos ha permitido observar la importancia que tienen los problemas de heurística en la vida real de manera práctica, y nos ha ayudado a aprender a implementar algoritmos de búsqueda heurística haciendo uso de heurísticas admisibles y bien informadas.