Práctica 2 - TSI

Laura Lázaro Soraluce

Ejercicio 1: Problema del coloreado de mapas

Primero planteamos el problema como un COP. Parto del problema similar a este que hay en las transparencias, un CSP, y le añado el array de precios cuya suma queremos minimizar. Este nuevo problema nos da una única solución, como era de esperar, pues sólo puede haber como máximo un valor que sea el **mínimo**. En nuestro caso, conseguimos colorear el mapa por un precio de 2900€.

Resultados:

total: 2900

"Argentina	": Naranja	350
"Bolivia	": Rojo	250
"Brasil	": Azul	450
"Chile	": Verde	100
"Colombia	": Verde	100
"Ecuador	": Rojo	250
"Guyana Francesa	": Verde	100
"Guyana	": Verde	100
"Paraguay	": Verde	100
"Peru	": Naranja	350
"Surinam	": Rojo	250
"Uruguay	": Rojo	250
"Venezuela	": Rojo	250

Al pasarlo ahora a CSP, queremos ver cuántas combinaciones hay tal que el valor de *total* sea 2900€. Es decir, buscamos todas las combinaciones posibles con las que se alcanza ese mínimo. Lo que hacemos es imponer que el valor de total sea 2900€ y pedir que se cumpla esa restricción. Esto se consigue para 8 soluciones diferentes.

Al calcular ahora cuántos colores como mínimo hacen falta para colorear el mapa, usamos la función card() que calcula cuántos valores diferentes se han usado dentro del array de regiones, y lo que queremos es minimizar este valor. Esto es un nuevo problema de COP y por lo tanto también tiene una única solución: 4 colores.

Ejercicio 2: Problema Lógico

En este caso tenemos un CSP pues solo queremos imponer unas restricciones y que se cumplan, pero no optimizar la solución. Yo he decidido modelarlo como un **array de longitud 4, donde cada posición puede tomar valores del 1 al 4**. Las cuatro posiciones del array simulan las letras a (posición 1), b (posición 2), c (posición 3) y d (posición 4). Los

valores que haya en esas cuatro posiciones del array simulan las letras w (representada por un 1), x (representada por un 2), y (representada por un 3), z (representada por un 4).
Resultados: 21 soluciones posibles

a<=>w	b<=>w	c<=>w	d<=>x
a<=>x	b<=>w	c<=>w	d<=>x
a<=>y	b<=>w	c<=>w	d<=>x
a<=>z	b<=>w	c<=>w	d<=>x
a<=>w	b<=>x	c<=>w	d<=>w
a<=>x	b<=>x	c<=>w	d<=>w
a<=>y	b<=>x	c<=>w	d<=>w
a<=>z	b<=>x	c<=>w	d<=>w
a<=>w	b<=>x	c<=>w	d<=>x
a<=>x	b<=>x	c<=>w	d<=>x
a<=>y	b<=>x	c<=>w	d<=>x
a<=>z	b<=>x	c<=>w	d<=>x
a<=>w	b<=>x	c<=>w	d<=>z
a<=>x	b<=>x	c<=>w	d<=>z
a<=>y	b<=>x	c<=>w	d<=>z
a<=>z	b<=>x	c<=>w	d<=>z
a<=>x	b<=>y	c<=>w	d<=>x
a<=>x	b<=>z	c<=>w	d<=>x
a<=>x	b<=>z	c<=>x	d<=>x
a<=>x	b<=>z	c<=>y	d<=>x
a<=>x	b<=>z	c<=>z	d<=>x

En la forma en que he resuelto este problema, he supuesto que varias variables de las 4 primeras (a, b, c, d) pueden estar relacionadas con la misma variable del 2º grupo (w, x, y, z).

Si supusiera que cada variable del primer grupo está relacionada con una del 2º grupo, y no se pueden repetir (he dejado una constraint comentada en el código para que se dé este caso), la única solución que obtengo es:

a<=>y	b<=>x	c<=>w	d<=>z

Ejercicio 3: Problema de los horarios

Resultados: Teniendo en cuenta que de las 5h de tutorías, cada profesor tiene 1h, obtengo 2 soluciones.

```
horario = [| 8: 9: 10: 11: 12: 13: 14:
           | 4, 4, 11, 10,  1,  1,  9
           | 4, 4, 11, 10,  1,  1,  7
           | 8, 8, 11, 10,  3,  3,  6
           | 5, 5, 11, 10,  3,  3,  2
           | 5, 5, 11, 10,  6,  7,  2|];
```

```
horario = [| 8: 9: 10: 11: 12: 13: 14:
          | 4, 4, 11, 10,  1,  1,  9
          | 4, 4, 11, 10,  1,  1,  7
          | 8, 8, 11, 10,  3,  3,  6
          | 5, 5, 11, 10,  3,  3,  2
          | 5, 5, 11, 10,  2,  6,  7|];
```

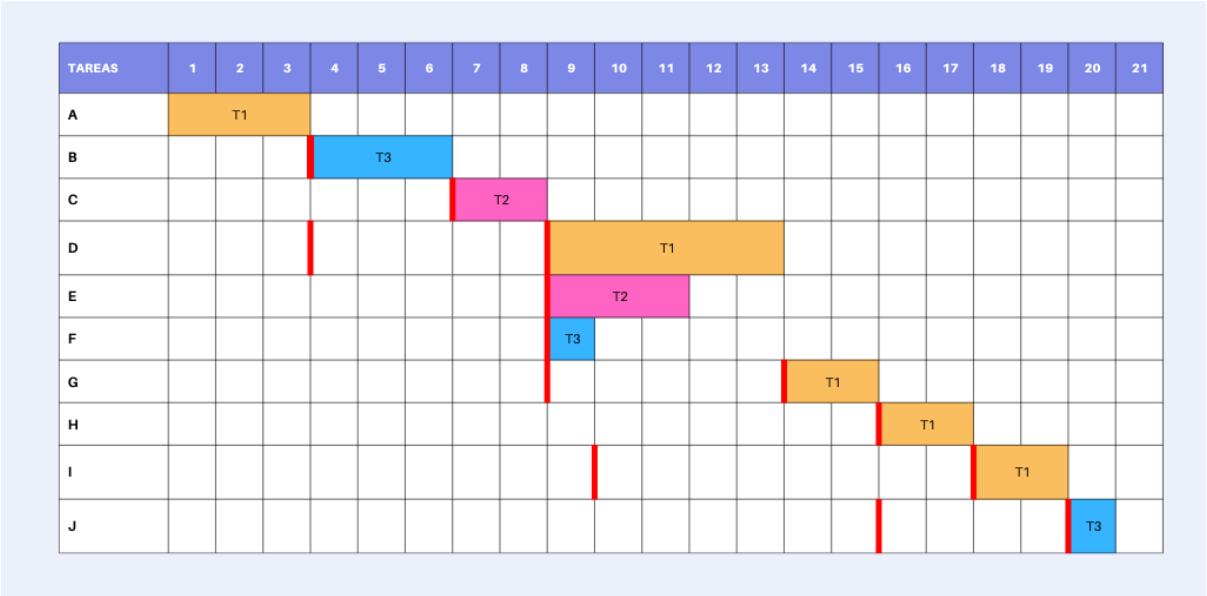
Soluciones simétricas: como he decidido codificarlo de manera que el array ASIGNATURAS tiene una posición para cada asignatura en lugar de una posición para cada bloque de cada asignatura, no hay soluciones simétricas. Si lo hubiese codificado de la segunda forma, serían soluciones simétricas aquellas en las que se intercambiasen dos bloques de la misma asignatura, pero he preferido evitar esto. De mi forma, si la asignatura X tiene 2 bloques de 1h, aparecen dos X en el horario, en lugar de X_1 y X_2, donde estos son los dos bloques de la asignatura. De esta segunda forma, el programa interpretaría diferente si primero se imparte X_1 y luego X_2, que si se imparte al revés, cuando semánticamente es lo mismo.

Ejercicio 4: Problema de asignación de tareas

Resultados apartado a): La duración mínima de la construcción de la casa con tres trabajadores es de 20 días. Es decir, el día 20 será el último día de construcción y el día 21 estarán libres todos los constructores y la casa terminada.

```
dia_inicio =    [|  1, 0,  0
                 |  0, 0,  4
                 |  0, 7,  0
                 |  9, 0,  0
                 |  0, 9,  0
                 |  0, 0,  9
                 | 14, 0,  0
                 | 16, 0,  0
                 | 18, 0,  0
                 |  0, 0, 20  |];
total = 20;
_objective = 20;
```

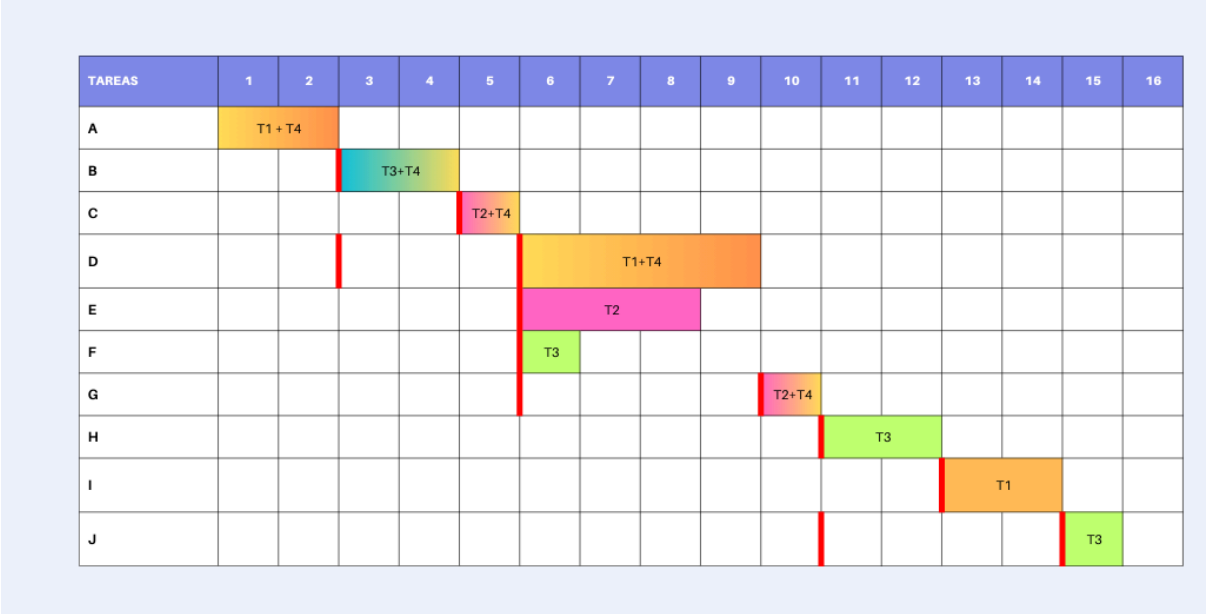
En esta tabla, las columnas representan los tres trabajadores, y las filas representan las diferentes tareas. El número que aparece en la posición [i,j] de la tabla, indica en qué día el trabajador *j* empezó a trabajar en la tarea *i*.



Las líneas rojas en cada fila indican que acabó una tarea que es necesaria para que pueda empezar la tarea de dicha fila.

Resultado apartado b): La duración mínima de la construcción de la casa con cuatro trabajadores es de 15 días. Es decir, el día 15 será el último día de construcción y el día 16 estarán libres todos los constructores y la casa terminada. Este apartado me tarda un poco más de los normal en ejecutarse (8s246ms), pero finalmente da resultado.

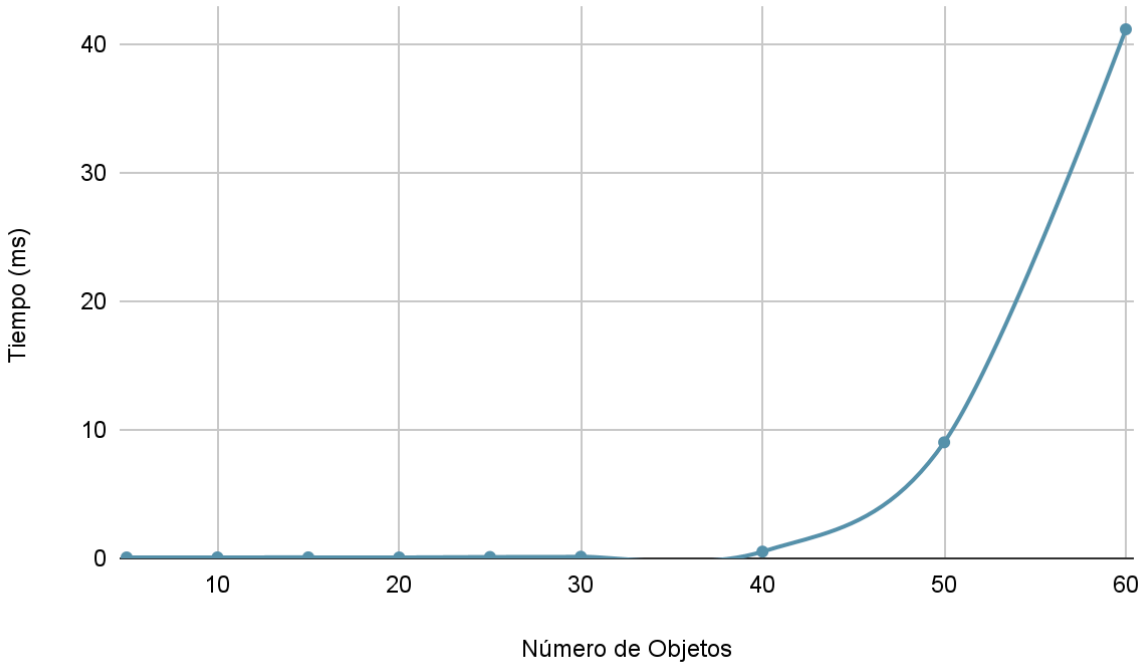
```
dia_inicio =    [|  1,  0,  0,  1
                 |  0,  0,  3,  3
                 |  0,  5,  0,  5
                 |  6,  0,  0,  6
                 |  0,  6,  0,  0
                 |  0,  0,  6,  0
                 |  0, 10,  0, 10
                 |  0,  0, 11,  0
                 | 13,  0,  0,  0
                 |  0,  0, 15,  0  |];
total = 15;
_objective = 15;
```



Las líneas rojas en cada fila indican que acabó una tarea que es necesaria para que pueda empezar la tarea de dicha fila.

Ejercicio 5: Problema de la mochila

Nº de objetos posibles	Nº de objetos incluidos en la mochila	Suma de las prioridades de los objetos de la mochila	Peso final de la mochila (en kgs)	Runtime
5	4	360	4.55	68msec
10	6	440	4.55	73msec
15	9	690	4.85	76msec
20	9	690	4.85	69msec
25	11	810	4.65	104msec
30	11	860	4.65	125msec
40	14	1065	4.95	523msec
50	16	1230	4.75	9s 36msec
60	17	1280	5.0	41s 209msec
70	—	—	—	>60sec



Basándome en los resultados de la tabla, diría que este problema no es altamente escalable, pues el tiempo de ejecución sube mucho para un aumento bastante pequeño del número de objetos incluídos. Fijándome también en la gráfica, veo que la función tiene forma exponencial en lugar de lineal, con lo que el aumento del runtime sería cada vez mayor para un aumento fijo del número de objetos. Llegaría un punto en el que la resolución de este problema sería prácticamente imposible, o tardaría un tiempo exagerado.