UNIVERSIDAD DE GRANADA

Grupo 1

Técnicas de los Sistemas Inteligentes — Práctica 2.Satisfacción de restricciones

Juan Mota Martínez
juanmotam@correo.ugr.es

3 de mayo de 2020

Índice general

§0.1: Ejercicio 1

Este ejercicio se nos pide encontrar una combinación entre las letras desde la A,E,S,T,K,R,N,I,D y los números entre el 0 y el 9, cada letra sólo puede tener un número asignado y viceversa, se nos pide encontrar una asignación se satifaga la siguiente sumatoria:

$$\begin{array}{r} +\text{TESTE} \\ +\text{FESTE} \\ +\text{DEINE} \\ \hline \text{KRAFTE} \end{array}$$

De forma que para la suma de la primera columa E tiene que ser igual al resultado de 3E mod 10. Esta primera suma se ha implementado muy fácilmente, ha bastando con crear las variables para cada letra y luego añadir una restricción para que se cumpla la asignación mostrada anteriormente. Las siguientes sumas son ligeramente más complicadas ya que también tenemos que tener en cuenta que tenemos que añadir las decenas de la suma anterior, de forma que en el hipotético caso de que en una suma el resultado fuese 15, guardaríamos 5 como valor y tendríamos que sumar 1 en la siguiente operación.

Se ha implementado la solución como un vector desde el 0 al 9 y cada una de sus componentes puede tener un valor entre 0 y 9. También se implementarion las letras como variables, aunque no fuese necesario, porque era mas fácil de entender y seguir las operaciones que se iban realizando, de forma que la letra T equivale a la componente 0 del vector, la letra E a las 1, etc... También se han introducido 4 variables nuevas para conservar los resultados de las sumas, de esta forma para la primera operación bastaría con las siguientes líneas:

```
1 constraint Resultado1 == (E+E+E);
2 constraint E == Resultado1 mod 10;
```

Nos guardaremos Resultado1 para futuras operaciones, por ejemplo la siguiente suma, Resultado2 equivale a T+T+N, pero en este caso T equivale al resultado en base 10 de esa suma más el cociente obtenido de dividir resultado1 entre 10. Las siguientes operaciones siguen la misma forma sumando los cocientes de resultados anteriores a las nuevas sumas. También para asegurarnos que se obtiene una asignación correcta se ha añadido la líena: constraint forall(i in 0..9)(count(Letras,i)<=1); que obliga a que todos los valores del vector sean distintos.

La solución generada nos devuelve la siguiente asignación:

$$\text{T} = 3;\ \text{E} = 0;\ \text{S} = 8;\ \text{F} = 6;\ \text{D} = 5;$$

$$\text{N} = 7;\ \text{I} = 9;\ \text{K} = 1;\ \text{R} = 4;\ \text{A} = 2;$$

§0.2: Ejercicio 2

El segundo ejercicio nos plantea un problema mucho más fácil de resolver, tenemos que encontrar un número de 10 dígitos que satisfaga que el primer dígito de x representa el número de 0 en x, el segundo el número de 1, etc...

mediante una array de 10 posiciones empezando por el 0, llamado Digitos y asegurándonos de que para cada posición el resultado de count(Digitos,numero), se corresponde con el número almacenado en Digitos[numero].

Se ha implementado usando 10 constraint forall i in (0..9)(count....) porque se descubrió más tarde como implementar dos bucles, se ha decidido mantener así porque devuelve la solución correcta. El número devuelto es 6210001000.

§0.3: Ejercicio 3

En este ejercicio se nos pide encontrar una asignación de clases que satisfaga las siguientes condiciones:

- Sólo existe un aula, disponible entre las 9 y as 15
- Esta sólo puede ser ocupada por un único profesor al mismo tiempo
- Cada clase dura una hora
- Existen restricciones en los horarios de los profesores.

Tenemos un array llamado Horario que representa a que hora empieza cada profesor su clase, por ejemplo si almacenamos 14 en la posición 1 eso quiere decir que el profesor 1 empieza su clase a las 14. Esto se ha implementado así porque todas las clases duran una hora de forma que la hora de inicio y fin sólo van a diferenciar por un dígito.

Tras esto nos aseguramos que por ejemplo para el profesor 1, el número almacenado tenga que ser mayor o igual que 11 y menor o igual que 14. Para evitar que dos profesores puedan tener clase al mismo tiempo se ha usando la restricción: forall(i in 9..14)(count(Horario,i)<=1), que nos garantiza que no hay dos valores iguales en el vector;

El resultado obtenido es: P1 empeiza su clase a las 14, P2 a las 12, P3 a las 13, P4 a las 10, P5 a las 11 y P6 a las 9.

§0.4: Ejercicio 4

Este ejercicio nos plantea una situación similar al problema anterior, asignar aulas a profesores con restricciones, pero en este caso tenemos 12 clases de una hora a asignar 4 aulas dispoinbles entre las 9 y las 13 y profesores que dan varias clases distintas.

Esta vez se ha decidido plantear las aulas y sus horas disponibles como una matriz de filas 1 a las 4 para representar el aula y con columnas de la 9 a las 12 para representar las horas de inicio de cada clase. tenemos 4 profesores los cuáles dan varias asignaturas, el número de asignaturas totales es 12, de forma que para resolver el problema lo hemos dividido en dos partes, asegurarnos que cada profesor da el número que le corresponde en sus horarios disponibles y siempre cumpliendo que no puede empezar dos clases al mismo tiempo, y la segunda parte asignar asignaturas a cada horario del profesor. De forma que hemos creado un segundo array con las mismas dimensiones que el primero, este segundo array recoge a qué hora empieza a dar clase un profesor y en que aula, primero se han tenido en cuenta las restricciones de tiempo, por ejemplo no se le puede asignar al profesor 2 una clase a las 10. Tras esto hemos tenido en cuenta que si un profesor empieza a dar clase a las 9 en el aula 1, no puede empezar otra al mismo tiempo en un aula distinta, esto se ha hecho mediante la restricción: constraint forall(c,i in 1..4, j in 9..12 where i!=c)(Profesores[c,j]!=Profesores[i,j]);

Ahora es necesario asegurarse que el profesor 1 aparezca 4 veces en la matriz porque ha de dar 4 clases, el segundo 2 veces, etc.. Una vez completada la matriz de profesores considerando los 0 como momentos en los que el aula está vacía, procedemos a asignar clases: para ello recorremos la matriz de profesores buscando un número en concreto e indicado que para ese número en la matriz de Aulas pueden asignarse ciertas clases, por ejemplo si nos encontramos un 1 en la posición 1,12 de la matriz de Profesores en la matriz de Aulas la posición 1,12 puede valer cualquiera de los siguientes números: 1,2,3 o 4. La codificación usada aparece en el código y al final de este apartado. Una vez se ha comprobado esto tenemos que asegurarnos que todas las asignaturas aparecen una única vez, como en el anterior usaremos la asignatura 0, la cuál si puede aparecer varias veces, para indicar que no hay clase.

La solución que nos devuelve el problema es la siguiente:

Aula1: FBDG4, IAG4, FBDG3, IAG3
Aula2: FBDG2, TSIG1, FBDG1, IAG2
Aula3: TSIG2, NADA, IAG1, NADA
Aula4: NADA, TSIG4, NADA, TSIG3

Codificación: 0-No hay clase 1-IA-G1 2-IAG2 3-TSIG1 4-TSIG2 5-FBDG1 6-FBDG2 7-TSIG3 8-TSIG4 9-FBDG3 10-FBDG4 11-IAG3 12-IAG4

§0.5: Ejercicio 6

Se trata de un ejercicio bastante simple, basta con introducir todas las restricciones para obtener la solución correcta, la forma en la que se ha implementado es la siguiente, se ha creado un array para cada condición: region, color, mascota, etc.. y en el array se almacena a que casa pertenece cada uno, y la posición del vector indica el tipo de mascota, bebida, etc.. la codificación, como en el ejercicio anterior aparece tanto en el código como al final de este ejercicio. El valor almacenado en cada vector indica a que casa pertenece, es decir si hay un 1 quiere deir que se trata de la casa más a la izquierda, un 3 se refiere a la casa central, etc... Ha bastado con introducir las restricciones para obtenr una solución que responde a las preguntas del ejercicio:

- La cebra están en la casa más a la derecha (5).
- Y se bebe agua en la primera casa por la izquierda (1).

Region: 1-Vasco 2-Catalan 3-Gallego 4-Navarro 5-Andaluz
Bebida: 1-te 2-leche 3-Zumo 4-café 5-Agua
Mascota: 1-Perro 2-caracoles 3-zorro 4-caballo 5-cebra
Color: 1-Roja 2-Verde 3-Blanca 4-Amarilla 5-Azul
Profesión: 1-Pintor 2-escultor 3-Diplomático 4-Violinista 5-medico

§0.6: Ejercicio 7

Para este ejercicio se nos piden encontra la asignación de tareas siguiendo las restricciones especificadas para tratar de completar la casa en el menor número de horas posibles, se ha creado un vector denomiando Orden de posiciones 1 a 9 para representar en que orden comienza cada tarea, por ejemplo, si se almacena un 1 en la posición 3 quiere decir que la tarea 3 es la primera en realizarse, como se pueden comenzar varias tareas al mismo tiempo hay valores en este vector que pueden repetirse, Exite otro vector denominado Tiempo que recoge el número de horas necesarias para completar cada tarea, como se nos indica que la tarea A tarda 7 horas en la posición 1 se almacena un 7, como en ejercicios anteriores la codificación aparece al final y en los comentarios. Ahora para considerar las restricciones que se nos indican, se obliga a que ciertos órdenes sean mayores que otros, para el caso de las tareas A y B, se nos dice que B tiene que realizarse siempre después de haberse finalizado A, luego existe una restricción para que Orden[2] siempre tenga que ser mayor que Orden[1]. El resto de restricciones se ha implementado de forma idéntica. Finalmente para realizar el recuento de horas

existe un último vector, el cuál acotamos primero usando una variable que igualamos al valor más alto del vector Orden, esta variable indica el órden en el que se realiza la última o últimas tareas.

Para realizar el recuento, buscamos primero los valores del vector de Orden que se repitan y tomamos el mayor de ellos guardándolos en un vector aparte denominado maxtiempos, tras esto buscamos los órdenes que no se repiten y guardanos el tiempo que les corresponde en el vector, una vez terminado usamos el operador de sumatoria e indicamos que se trata de un problema de minimización, el tiempo mínimo que se tarda en construir la casa son 22 horas.

Codificación: 1-A,2-B,3-C,4-D,5-E,6-F,7-G,8-H,9-I

§0.7: Ejercicio 10

El último ejercicio Nos pide que resolvamos el problema de la mochila, se ha usado un implentación sencilla, empleando dos vectores para representar las importancia y peso de los objetos. Un ultimo vector representa la mochila que almacena 0 o 1, un 0 indica que el objeto no se ha toma mientras que un 1 indica que el objeto está presente en la mochila. Como se nos indica que el máximo peso que se puede soportar es 275, se ha introducido una variable denominada peso, peso se calcula como la sumatoria del vector de pesos cuyos índices equivalen a 1 en el vector mochila, se hace lo mismo para el calculo de la valoración de la mochila y finalmente se introduce la restricción de que el peso no puede superar 275, el problema devuelve que la valoración máxima que se puede obtener es 705, con un peso de 274.