# Memoria Práctica 2

Álvaro Beltrán

May 2, 2020

Grupo 3, Jose Ángel Segura Muros

# 1 Introducción

En esta práctica se pide al estudiante resolver la siguiente relación de problemas usando MiniZinc.

# 2 Ejercicios

## 2.1 Ejercicio 1

Para el ejercicio 1 he creado dos vectores de 10 posiciones cada uno, En el primero, llamado letras, he almacenado las letras que hay que codificar. El segundo, llamado n, es el vector para la solución donde cada componente es el número de la letra en esa posición.

$$letras = [\text{"}A\text{"},\text{"}E\text{"},\text{"}F\text{"},\text{"}I\text{"},\text{"}S\text{"},\text{"}T\text{"},\text{"}N\text{"},\text{"}K\text{"},\text{"}R\text{"},\text{"}D\text{"}]$$

Para las restricciones he definido dos:

- Que sean todas las componentes de n diferentes.
- La siguiente operación para que respete la suma del enunciado:

$$\begin{aligned} &3 * n[2] + 20 * n[6] + 200 * n[5] + 3000 * n[2] + 10000 * n[6] \\ &+10000 * n[3] + 10000 * n[10] + 10 * n[7] + 100 * n[4] \\ == &\ n[2] + 10 * n[6] + 100 * n[3] + 1000 * n[1] + 10000 * n[9] + 100000 * n[8] \end{aligned}$$

**solución.**
"A" = 6 | "E" = 5 | "F" = 3 | "I" = 8 | "S" = 2 | "T" = 9 | "N" = 0 | "K" = 1 | "R" = 7 | "D" = 4

## 2.2 Ejercicio 2

En el ejercicio 2 he creado un vector de 10 posiciones que representa el número de 10 dígitos. En la restricción he usado un forall para recorrer todos los dígitos, y con la función count he contado las veces que está ese número indicado en el vector. Escribiendo como restricción esto: POS=(1..10)

```
1 constraint forall(i in POS)(n[i]==count(n,i-1));
```

**solución.** 6210001000

## 2.3 Ejercicio 3

En el ejercicio 3 he creado un vector de 6 posiciones que representa el horario (hay seis horas a las que asignar un profesor). Cada hueco del vector puede ser rellenado por un número del 1 al 6 que representa cada uno de los profesores.

En cuanto a las restricciones he definido que sean todas las posiciones del vector diferentes dado que solo puede impartir un profesor en la clase. Y he definido las restricciones horarias traduciendo a la nomenclatura ya definida que tal número (entre 1 y 6) no puede estar en tal casilla del vector.

**solución.**

09:00-10:00 $\longrightarrow$ PROF-6
10:00-11:00 $\longrightarrow$ PROF-4
11:00-12:00 $\longrightarrow$ PROF-5
12:00-13:00 $\longrightarrow$ PROF-2
13:00-14:00 $\longrightarrow$ PROF-3
14:00-15:00 $\longrightarrow$ PROF-1

## 2.4 Ejercicio 4

En este problema he tomado la decisión de asignar a cada asignatura de un grupo por un número, de esta forma : [”IA-G1”,”IA-G2”,”IA-G3”,”IA-G4”,”TSI-G1”,”TSI-G2”,”TSI-G3”,”TSI-G4”,”FBD-G1”,”FBD-G2”,”FBD-G3”,”FBD-G4”,”hueco”,”hueco” ,”hueco”,”hueco”] la posición de este vector es el número asociado a cada asignatura, teniendo en cuenta que tenemos en el horario 4 huecos para asignar (13,14,15,16). Además tenemos dos matrices 4x4 una para la asignación de los grupos (con valores 1..16) y otra para la asignación de los profesores (con valores 1..8 teniendo en cuenta que los valores 5,6,7,8 son huecos).

En cuanto a las restricciones:

- Para que cada asignatura se de una vez por semana y no más de una hora, he definido la restricción de que en la matriz asociada a los grupos todos los números deben ser diferentes.
- Para que cada grupo reciba docencia de una única asignatura. He implementado una restricción que recorre todas las casillas de la matriz de grupos saltándose si la asignación es 13,14,15,16 (huecos) y comprobando que no hay mas asignaturas de ese grupo en esa fila usando el operador módulo 4.
- He creado restricciones para satisfacer la tabla de los profesores.
- Para que un profesor no pueda estar en dos clases a la vez he implementado una restricción para que todos los valores de la matriz de profesores por filas sean diferentes.
- Por último he tenido que definir una restricción para que si he puesto un hueco en la matriz de asignaturas esté el respectivo hueco en la matriz de profesores.

Obligando a que se den las parejas porf-5 <-> asig-13, porf-6 <-> asig-14, porf-7 <-> asig-15, porf-8 <-> asig-16

**solución.**

| | Aula 1 | | Aula 2 | | Aula 3 | | Aula 4 | |
|---|---|---|---|---|---|---|---|---|
| 1h | nada | | Asig-9 | Prof-2 | Asig-7 | Prof-3 | Asig-2 | Prof-1 |
| 2h | nada | | Asig-8 | Prof-3 | Asig-3 | Prof-4 | Asig-1 | Prof-1 |
| 3h | nada | | nada | | Asig-12 | Prof-3 | Asig-6 | Prof-1 |
| 4h | Asig-11 | Prof-3 | Asig-10 | Prof-2 | Asig-5 | Prof-1 | Asig-4 | Prof-4 |

## 2.5 Ejercicio 5

Para este problema he creado tres matrices: **asig** de 6x5 con valores del 1 al 10 (cada número representa una asignatura y el número 10 es el recreo), **prof** de 6x5 con valores del 1 al 5 (cada número es un profesor y el número 5 es el profesor asociado al recreo) y por último la matriz **bloques** de 3x5 que representa los bloques de 2h disponibles en el horario con valores del 0 al 9 siendo 0 que no se imparte clase de dos horas y del 1 al 9 que asignatura de dos horas se imparte.
En cuanto a las restricciones programadas tenemos:

- Primero he obligado a que en la cuarta fila de las matrices asig y prof estén rellenas por los números 10 y 5 respectivamente.
- He contado el número de asignaturas asignadas en la matriz por que cada asignatura tiene un número semanal de horas asignadas.
- He definido una serie de restricciones sobre los bloques:
  - No pueden estar los números 2,6,7 y 9; ya que estos no se pueden impartir en bloques de dos horas.
  - Si el primer bloque de dos horas esta ocupado el segundo debe estar desocupado y viceversa, esto se debe a que el primer y segundo bloque comparten una hora.
  - El número de bloques desocupados es de 6.
- Para definir que no puede haber más de un bloque de una asignatura diaria he obligado a que las columnas de la matriz bloques sean distintas, excepto el cero y que en la matriz asig no pueden aparecer los valores 2,6,7 y 9 más de una vez por columnas.
- Para que los bloques se correspondan con la matriz asig, he tenido que hacer una restricción para que cada bloque signifique 2h de la matriz asig consecutivas.
- Por último, las restricciones sobre profesores:

- He definido restricciones para que los profesores den las asignaturas que están definidas en el enunciado.
- Para que los profesores no tengan dos bloques el mismo día, he hecho que las asignaturas 1-3 y 4-5 no puedan coincidir en las columnas de bloque. Y que el profesor 3 no puede aparecer más de una vez por día en la matriz de profesores.

Por último, decir que el programa ejecuta en 22 seg si no defino outputs. **solución.**

| | Lunes | | Martes | | Miércoles | | Jueves | | Viernes | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1h | A8 | Prof-4 | A3 | Prof-1 | A2 | Prof-4 | A6 | Prof-3 | A2 | Prof-4 |
| 2h | A8 | Prof-4 | A3 | Prof-1 | A6 | Prof-3 | A1 | Prof-1 | A1 | Prof-1 |
| 3h | A9 | Prof-3 | A7 | Prof-4 | A7 | Prof-4 | A1 | Prof-1 | A1 | Prof-1 |
| 4h | Recreo | | | | | | | | | |
| 5h | A5 | Prof-2 | A4 | Prof-2 | A3 | Prof-1 | A4 | Prof-2 | A5 | Prof-2 |
| 6h | A5 | Prof-2 | A4 | Prof-2 | A3 | Prof-1 | A4 | Prof-2 | A5 | Prof-2 |

## 2.6 Ejercicio 6

En el ejercicio 6 he usado 5 vectores, cada uno referente a los gustos de los propietarios de las casas. Cada posición del vector ubica una casa, cada vector puede tomar valores desde el 1 al 5 representando cada uno de los diferentes grupos.
En cuanto a las restricciones tenemos:

- Cada uno de los vectores tiene todos sus números diferentes, por que ninguna persona comparte nada con nadie.
- Luego he creado una restricción por cada punto que plantea el enunciado.

**solución.**

| | Casas | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| colores | amarilla | azul | roja | blanca | verde |
| personas | andaluz | navarro | vasco | catalán | gallego |
| animales | zorro | caballo | caracoles | perro | **cebra** |
| trabajos | diplomático | médico | escultor | violinista | pintor |
| bebidas | **agua** | té | leche | zumo | café |

## 2.7 Ejercicio 7

Para el ejercicio 7 he usado un vector en el cual cada coordenada representa el día en el que empieza la tarea x por orden alfabético. Además he tomado como que el día en que

termina una tarea no puede empezar otra que la tiene como predecesora. Para facilitar la construcción de las restricciones he creado un vector con la duración de las tareas (en días).
En cuanto a las restricciones tenemos:

- Las restricciones oportunas para satisfacer la tabla, de forma que una tarea siempre tiene que empezar después del día que empezó su predecesora más la duración de la predecesora.
- Además una restricción para que la variable duracionTotal tenga almacenado cuando termina la construcción.

**solución.** duración total=18

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| Dia de Inicio | 1 | 8 | 11 | 8 | 16 | 16 | 16 | 8 | 17 |

## 2.8 Ejercicio 8

Para el ejercicio 8, como modificación del ejercicio 7 que es, he usado todo lo del ejercicio 7 añadiendo dos contenedores y unas restricciones. El primero es un vector llamado numT que almacena los trabajadores necesarios para cada tarea. El segundo es una matriz que hace de calendario para los trabajadores indicando cuantos trabajadores y en que día deben trabajar. Esta matriz se compone de 25 días, ya que manualmente encontré una solución en 25 días. (la solución que encuentra el problema es de 22 días).
En cuanto a las restricciones he añadido a las del ejercicio 7 estas:

- Una restricción para que los trabajadores de una tarea trabajen únicamente en esa tarea los días necesarios hasta terminarla
- Otra restricción para que trabajen el número necesario de trabajadores que necesita cada tarea.
- La última es para que nunca haya más de 3 trabajadores en un mismo día.

**solución.**

## 2.9 Ejercicio 9

Para el ejercicio 9, como modificación del ejercicio 8 que es, he usado todo lo del ejercicio 8 añadiendo dos contenedores nuevos, una matriz durTrabajadores que almacena lo que tarda cada trabajador en completar una determinada tarea. Y un vector trabajadores donde almaceno que trabajador hace cada tarea. En el calendario ahora almaceno que trabajador está haciendo la tarea indicada. He reducido los días del calendario a 15 para que tarde menos en encontrar la óptima.
En cuanto a las restricciones, he añadido una serie de nuevas restricciones:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 2 | 2 | 2 | 2 | 2 | 2 | 2 | | | | | | | |
| B | | | | | | | | | | | | | | |
| C | | | | | | | | | | | | | | |
| D | | | | | | | | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| E | | | | | | | | | | | | | | |
| F | | | | | | | | | | | | | | |
| H | | | | | | | | | | 1 | 1 | 1 | | |
| I | | | | | | | | | | | | | | |

| | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|
| A | | | | | | | | |
| B | | 3 | 3 | 3 | | | | |
| C | | | | | 2 | | | |
| D | 2 | | | | | | | |
| E | | | | | | | 1 | 1 |
| F | | | | | | 2 | | |
| H | | | | | | 1 | | |
| I | | | | | | | 2 | 2 |

- Una restricción para que si el trabajador i hace la tarea j. entonces la duración de la tarea debe ser la que tarde el trabajador i en hacerla.
- Otra para que el trabajador esté colocado en el calendario el número de veces que indique duración.
- La duración de una tarea debe ser la respectiva a algún trabajador.
- El trabajador debe estar puesto en el calendario todos los días desde que empieza la tarea hasta que la acaba.
- Cada trabajador no puede hacer más de una tarea al día.

**solución.** (Calendario) duracionTotal=12

### 2.10 Ejercicio 10

Por último, en el ejercicio 10 he definido: un vector de pesos donde almaceno los pesos de los objetos, un vector de preferencias donde almaceno las preferencias de los objetos, dos enteros peso y pref donde almaceno el peso y la preferencia de la mochila, Maximizo el valor de pref. Los huecos de la mochila los he prefijado a 8 porque ya con 9 huecos tardaba mucho.
En cuanto a las restricciones:

- Una restricción para que la variable peso sea la suma de los pesos de la mochila.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 1 | 1 | 1 | 1 | | | | | | | | |
| B | | | | | 1 | 1 | 1 | | | | | |
| C | | | | | | | | 2 | | | | |
| D | | | | | | | | 1 | 1 | | | |
| E | | | | | | | | | | 2 | 2 | |
| F | | | | | | | | | | 3 | | |
| G | | | | | | | | | | 1 | | |
| H | | | | | 2 | 2 | 2 | | | | | |
| I | | | | | | | | | | | 1 | 1 |

- Una restricción para que la variable pref sea la suma de las preferencias de la mochila.
- Una restricción para que no pueda coger el mismo objeto más de una vez.
- La última restricción para que peso no sea mayor que 275.

**solución.** Peso=274 Preferencia=705 Tiempo:1min 20s

| mochila | | | | | | | |
|---|---|---|---|---|---|---|---|
| Protector Solar | Queso | Azúcar | Sandwich | Agua | Compás | Mapa | Nada |

# 3 Conclusión

Práctica interesante, a la cual para facilitar la resolución de los problemas hay que acostumbrarse al lenguaje que en ella usamos para programar, que de primeras se hace costoso.