PRÁCTICA 1

Programación con Restricciones

Alumno:

Luis Pozas Palomo

1. Restricciones codificadas

Las **restricciones paramétricas** que detectan alguna discrepancia en los parámetros de entrada y que me permiten parar la ejecución antes de empezar a buscar son:

RESTRICCIONES PARAMÉTRICAS	
`D > 0`	Los días a planificar deben ser mayor que 0.
`T > 0`	El número de trabajadores debe ser mayor que 0.
`C > 0`	Cada trabajador pertenece como mínimo a una categoría.
`A >= 0`	El número de trabajadores afines debe ser un número positivo.
`(N1+N2+N3) <= T`	Debe haber trabajadores suficientes para cubrir todos los turnos.
`N1>0 /\ N2>0 /\ N3>0`	Cada turno debe tener como mínimo un trabajador asignado.
`MaxDT > 0`	El máximo de días trabajados consecutivos debe ser como mínimo uno.
`D-(MaxDT-1) > 0`	El máximo de días trabajados consecutivos no debe sobrepasar el número de días planificados.
`MaxDL > 0`	El máximo de días libres consecutivos debe ser como mínimo uno.
`D-(MaxDL-1) > 0`	El máximo de días libres consecutivos no debe sobrepasar el número de días planificados.
`MinDT < D /\` `MinDT > 0`	El mínimo de días trabajados debe ser menor que los días planificados y mayor que 0
`forall(t in 1..T)` **`(MinDT - diasLibres[t]) > 0 /\ diasLibres[t] >= 0`**	Los días libres que piden los trabajadores deben ser positivos y la diferencia con el mínimo número de días trabajados debe ser mayor que uno.

Antes de comenzar a codificar las restricciones, debo de buscar una representación de la solución, para ello voy a utilizar una matriz en la que las filas representan los trabajadores y las columnas los días a planificar. Las **restricciones del problema** que he codificado y me permiten buscar una solución son:

1. Cada turno tiene el número de trabajadores (N1, N2 o N3) que le corresponde:
Para todos los días a planificar, la suma de cada turno de todos los trabajadores es igual al número de trabajadores de cada turno.

2. Un trabajador solo puede estar en un turno cada día.
La representación de la solución del problema no permite asignar más de un turno a cada trabajador.

3. Dado un número MaxDT, garantizar que nadie trabaja MaxDT días consecutivos.
Para todos los trabajadores y para todos los días a planificar, la suma por cada MaxDT días consecutivos que tengan algún turno asignado debe ser menor que MaxDT.

4. Dado un número MaxDL, garantizar que nadie tiene MaxDL días libres consecutivos.
Para todo trabajador y días, la suma por cada MaxDL días consecutivos que sean libres deben ser menor que MaxDL.

5. Dado un número MinDT, garantizar que todos trabajan como mínimo MinDT en los D días.
Para todo trabajador, la suma de los días que tengan algún turno asignado debe ser mayor o igual a MinDT descontando los días libres pedidos por cada trabajador.

6. Un trabajador no puede hacer el último turno de un día y el primero del día siguiente dos veces seguidas en cuatro días consecutivos.
Para cada trabajador y para cada cuatro días, si el primer día tiene el último turno asignado, no puede tener el día siguiente el primer turno asignado, y los dos días siguientes debe cumplirse la misma restricción.

7. Dada una serie de parejas de trabajadores afines y un número A, cada trabajador de un turno tiene que estar con al menos A trabajadores afines en ese turno.
Para cada día, si el número de trabajadores por turno es mayor que uno, entonces busco por cada trabajador el número de trabajadores que son afines a este y tienen su mismo turno, finalmente este valor debe ser mayor o igual que A.

8. Siendo C el número de categorías y dadas las categorías de los trabajadores. En cada turno debe haber al menos un trabajador de cada categoría. Podéis generalizar esta restricción a que se indique exactamente (o como mínimo) cuántos de cada categoría tiene que tener cada turno y tenerlo distinto para cada uno de los tres turnos.
Para poder implementar esta restricción, voy a comparar para cada día, si el cardinal del conjunto de categorías de los trabajadores de ese día concreto por cada tipo de turno es mayor o igual a C.

Las **funciones objetivo del problema** que permiten optimizar la solución son:

1. Añadid la posibilidad de que los trabajadores pidan días que no quieren trabajar. Tened en cuenta que estos días no cuentan con la restricción de días consecutivos libres y que hay que descontarlos de MinDT. Convertid el problema en un problema de optimización, minimizando el número de veces que hacemos trabajar a alguien cuando no quiere. Pensad en formas de conseguir que los incumplimientos se distribuyan entre todos.

Primero he planteado la restricción en la que para cada trabajador debe existir una secuencia de días libres consecutivos (vacaciones) que ha elegido.

Restricción para plantear este objetivo:

```
constraint forall(t in 1..T where diasLibres[t] != 0 )
                  (exists (d in 1..(D - (diasLibres[t]-1)))
                        ((sum(k in d..(d + (diasLibres[t]-1)))
                              (bool2int(asig[t,k] = 0))) = diasLibres[t]));
```

Para buscar el problema de minimización planteado se puede transformar en uno de maximización siempre y cuando maximizamos el número de trabajadores que consiguen sus días libres consecutivos que han pedido. Por lo tanto debo maximizar la suma por cada trabajador que haya conseguido sus días libres pedidos.

```
solve maximize sum(t in 1..T where diasLibres[t] != 0 )
            (bool2int((exists (d in 1..(D - (diasLibres[t]-1)))
                  ((sum(k in d..(d + (diasLibres[t]-1)))
                        (bool2int(asig[t,k] = 0))) = diasLibres[t]))));
```

2. Añadid la posibilidad de que los trabajadores pidan qué turno no quieren (único o para cada día), si es que tienen alguna preferencia. Convertid el problema en un problema de optimización, minimizando el número de veces que incumplimos la petición. Pensad en formas de conseguir que los incumplimientos se distribuyan entre todos.

Primero he planteado la restricción en la que por cada trabajador y día planificado, el turno asignado un cierto día debe ser distinto al que ha pedido ese día el trabajador.

Para buscar el problema de minimización, primero lo voy transformar a buscar el número de trabajadores máximos que han conseguido pedir el turno que no quieren, de esta manera estoy beneficiando a unos trabajadores antes que a otros ya que podrían repartirse los turnos por trabajadores para que todos puedan conseguir el turno que han pedido y que no quieren.

Para encontrar la felicidad entre trabajadores, en lugar de repartir los turnos por trabajadores solamente, voy a repartirlos por trabajadores y por días, de esta manera según la distribución de turnos planificados podemos encontrar una planificación en la que todos los trabajadores

tengan la oportunidad de conseguir el turno que han pedido y que no quieren. Es cierto que los trabajadores que han pedido los días al comienzo de la planificación tienen más prioridad que los que los piden los últimos días, ya que como bien he comentado anteriormente estos se reparten por días, comenzando desde el día uno.

```
constraint forall(t in 1..T, d in 1..D where turnoLibre[t, d] != 0 )
(asig[t, d] != turnoLibre[t, d]);
```

Busco satisfacer completamente a cada trabajador.

```
solve maximize sum(t in 1..T)
                  (bool2int (forall(d in 1..D where turnoLibre[t, d] != 0 )
                              (asig[t, d] != turnoLibre[t, d])));
```

Busco encontrar la felicidad entre los trabajadores, voy asignando por días.

```
solve maximize sum(t in 1..T, d in 1..D where turnoLibre[t, d] != 0)
                  (bool2int(asig[t, d] != turnoLibre[t, d]));
```

2. Tiempos

Para los tres conjuntos de datos de prueba que he proporcionado he obtenido los siguientes resultados:

- "**data1_pr1**" en el que planifico 30 días, con 4 trabajadores en el que se trabaja 5 días consecutivos con un máximo de 2 días libres consecutivos, el mínimo de días trabajados es 22 y en cada turno se precisa de un único trabajador.
- "**data2_pr1**" en el que planifico 15 días, con 15 trabajadores en el que se trabaja 6 días consecutivos con un máximo de 7 días libres consecutivos, el mínimo de días trabajados es 2 y en cada turno se precisa de 2 trabajadores.
- "**data3_pr1**" en el que planifico 15 días, con 15 trabajadores en el que se trabaja 6 días consecutivos con un máximo de 7 días libres consecutivos, el mínimo de días trabajados es 2 y en cada turno se precisa de 2 trabajadores.

1) La búsqueda de una solución **sin** optimización, es la siguiente:

   Running practica1.mzn with **data1_pr1**.dzn
   0 3 3 2 3 3 0 3 3 2 3 0 3 2 0 1 3 0 1 2 1 1 2 0 3 3 3 2 3 0
   3 0 2 0 1 0 3 2 2 0 1 2 1 1 3 0 1 2 2 3 0 2 1 2 1 0 2 1 2 3
   2 2 0 3 2 1 2 0 0 3 2 1 0 3 2 3 2 1 0 1 2 0 0 1 2 1 1 3 0 2
   1 1 1 1 0 2 1 1 1 1 0 3 2 0 1 2 0 3 3 0 3 3 3 3 0 2 0 0 1 1

   Finished in 6s 258msec

   Running practica1.mzn with **data2_pr1**.dzn
   2 0 3 3 0 3 3 3 3 2 0 1 1 1 3

0 1 0 0 0 0 0 0 1 0 3 0 2 2 0
0 1 0 0 0 0 0 0 1 0 3 3 2 2 0
2 0 1 1 0 3 3 2 2 1 0 0 0 0 0
0 0 0 3 0 0 1 2 0 3 0 1 0 0 3
0 0 0 0 0 0 0 3 3 3 0 0 0 0 0
0 0 0 0 1 0 1 0 0 0 0 3 3 1 2
0 0 0 0 2 1 0 0 0 0 1 0 0 0 0
3 0 0 2 3 2 0 0 0 0 2 2 0 0 1
1 2 1 0 0 0 0 0 0 0 1 0 0 0 0
0 3 0 0 1 0 0 0 0 0 2 0 0 0 0
3 3 3 2 2 0 2 0 2 2 0 0 3 3 1
0 0 2 0 0 1 2 0 0 1 0 2 1 3 2
0 2 2 1 3 2 0 1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 1 0 0 0 0 0 0 0

Finished in 5s 503msec

Running practica1.mzn with **data3_pr1**.dzn
3 3 3 0 0 1 3 0 0 0 2 2 0 1 2 2 1 2 2 1 1 2 1 2 1 1 3 2 3 2
0 0 3 0 0 0 2 0 0 0 3 0 3 2 2 0 3 0 0 0 0 1 0 0 1 0 0 0 0 0
0 0 2 3 0 0 3 0 0 0 0 1 2 3 0 3 0 0 0 0 0 3 0 0 0 3 2 1 0 0
0 0 1 2 0 0 2 0 0 0 3 0 0 0 3 3 2 1 3 2 3 3 3 2 3 3 2 3 2 3
2 0 0 0 0 0 3 0 0 0 3 2 0 1 3 0 2 1 1 1 1 3 3 0 0 0 0 3 0 0
2 0 0 0 0 3 0 0 1 0 0 3 0 0 0 0 0 2 3 3 2 0 2 1 2 0 0 0 1 3
0 0 2 0 3 2 0 3 3 2 1 1 2 0 1 3 0 0 1 0 0 1 0 3 3 2 1 1 3 1
0 3 0 0 2 3 0 3 2 1 2 0 1 0 0 1 3 3 3 2 3 0 1 3 3 3 3 3 2 3
0 2 3 0 3 3 0 1 1 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3 0 0 0 1 2 0 2 3 3 0 3 3 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 0 2 2 0 0 3 0 1 0 3 0 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3 0 0 3 0 1 0 1 3 3 0 0 3 3 0 1 1 3 2 3 3 2 3 3 2 2 1 2 1 1
0 2 0 3 3 0 0 2 2 2 1 0 1 3 0 2 3 3 0 3 2 0 2 1 0 1 3 0 3 2
1 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Finished in 54s 458msec

2) Añadiendo la posibilidad de que los trabajadores pidan días que no quieren trabajar se añade al modelo el parámetro **"diasLibres"**, el cual es un array de trabajadores en el que cada posición indica los días que no quieren trabajar, he considerado estos días como vacaciones y por tanto se asignan de forma consecutiva.

Running practica1.mzn with **data1_pr1**.dzn
1 1 1 0 1 3 2 1 1 0 1 0 0 1 1 0 1 0 1 1 3 3 0 2 1 3 3 2 0 1
0 0 0 1 3 2 3 0 3 2 2 2 3 0 3 2 2 3 2 0 1 0 2 1 2 0 2 1 2 3
3 3 3 3 2 0 0 3 2 1 0 1 1 3 2 1 0 1 0 2 2 1 3 0 3 2 1 3 3 0
2 2 2 2 0 1 1 2 0 3 3 3 2 2 0 3 3 2 3 3 0 2 1 3 0 1 0 0 1 2

Finished in 5s 177msec

Running practica1.mzn with **data2_pr1**.dzn

2 0 3 0 0 0 0 3 0 0 0 0 0 0 0
0 0 1 0 2 0 0 0 3 2 0 2 1 3 0
0 0 1 0 2 3 0 0 3 2 0 2 1 3 0
0 0 0 0 0 0 0 2 0 0 1 0 0 0 0
2 2 0 3 0 0 0 2 0 0 3 0 0 0 0
1 1 2 1 0 0 2 3 0 0 1 0 2 0 3
0 2 0 1 0 0 0 0 0 3 3 0 0 0 0
3 0 2 2 3 0 2 0 2 3 2 0 2 1 0
0 3 0 0 0 2 3 0 1 1 0 3 3 2 1
0 1 0 0 1 2 1 0 0 0 0 0 0 0 2
0 3 0 3 0 1 1 0 0 1 0 3 0 0 1
3 0 0 0 3 0 0 0 1 0 0 1 0 1 3
0 0 0 2 1 3 3 0 2 0 2 1 3 2 2
0 0 3 0 0 1 0 1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 1 0 0 0 0 0 0 0

Finished in 8s 666msec

3) Para dar la posibilidad de que los trabajadores pidan su turno odiado para cada día, he añadido al modelo el parámetro **"turnoLibre",** el cual es una matriz en el que las filas representan a los trabajadores y las columnas los días, el valor que hay en cada celda es un natural entre 0 y 3 indicando si no odian ningún turno (0), si odian el turno de mañana(1), si odian el turno de tarde (2) o si odian el turno de noche (3).

Running practica1.mzn with **data1_pr1**.dzn

2 0 0 2 2 2 3 0 3 3 2 3 0 3 3 2 3 0 3 3 0 3 2 0 3 2 1 3 2 0
3 3 3 3 0 1 2 3 2 1 0 1 3 2 0 1 2 2 0 2 2 1 0 3 2 1 3 2 0 2
0 2 2 1 3 0 1 2 1 2 3 0 2 0 2 3 0 3 2 0 3 2 3 2 0 0 2 1 3 3
1 1 1 0 1 3 0 1 0 0 1 2 1 1 1 0 1 1 1 1 1 0 1 1 1 3 0 0 1 1

Finished in 3s 435msec

Running practica1.mzn with data2_pr1.dzn

0 0 0 0 0 0 2 3 0 3 2 0 2 3 0
0 1 0 2 1 0 0 0 2 0 0 2 0 2 0
0 1 0 2 1 0 0 0 2 1 0 2 0 2 0
2 0 1 0 0 3 2 2 0 0 0 0 0 0 0
3 3 0 3 0 3 3 2 0 0 0 1 3 1 3

2 0 3 0 3 0 0 3 0 0 0 0 0 0 0
3 3 0 3 0 2 0 0 0 0 0 3 3 0 1
0 0 1 0 0 2 0 0 0 1 1 0 2 0 3
0 2 2 1 2 1 1 0 1 2 3 0 1 0 0
0 0 3 0 0 0 0 0 3 0 1 1 0 0 0
0 2 2 0 0 0 0 0 3 0 0 3 0 1 2
0 0 0 0 3 0 1 0 0 3 3 0 1 0 1
0 0 0 1 2 1 3 0 1 2 2 0 0 3 2
1 0 0 0 0 0 0 1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 1 0 0 0 0 0 0 0

Finished in 9s 766msec