Práctica 1: Fábrica

Juan Pablo Fernández de la Torre

Arturo Gómez Izquierdo

Índice

1. Introducción 0
2. Problema 1
    2.1. Restricciones Básicas 1
    2.2. Restricciones de Optimización 2
    2.3. Restricciones extras 3
5. Juegos de Prueba 4
7. Opciones de Búsqueda Probadas 23

1. Introducción

En esta práctica se desarrolla un modelo de programación con restricciones en MiniZinc para la planificación de turnos en una fábrica que opera continuamente. Se presentan dos versiones del problema: una de satisfacción de restricciones y otra de optimización.

2. Problema

La fábrica tiene 3 turnos: mañana, tarde y noche, y cada turno necesita “N1”, “N2” y “N3” trabajadores respectivamente. Se debe asignar turnos a “T” trabajadores durante “D” días, cumpliendo una serie de restricciones. Se presentan dos posibles implementaciones de la solución:

1. Indicar qué turno tiene asignado cada trabajador cada día.
2. Indicar qué trabajadores están en cada turno cada día.

Siendo la primera la escogida para solucionar este problema, puesto que facilita la comprensión de la salida y la estructura de la solución es más sencilla de interpretar.

2.1. Restricciones Básicas

A continuación, se detallan las restricciones implementadas en la solución:

1. Cada turno tiene el número exacto de trabajadores requerido ("N1", "N2", "N3").

Para ello se recorre todos los turnos de todos los días comprobando que la cantidad de trabajadores que trabajan en cada turno coincide con el número dado.

2. Un trabajador solo puede estar en un turno por día.

Esta restricción viene implícita en la representación de la solución porque cada trabajador tiene un único turno en un día.

3. Ningún trabajador trabaja más de "MaxDT" días consecutivos.

Para cada trabajador, se revisa cada posible ventana de MaxDT días, y se exige que haya al menos un día de descanso dentro de esa ventana.

4. Ningún trabajador tiene más de "MaxDL" días libres consecutivos.

Al igual que la restricción anterior pero comprobando que en ese rango trabaje al menos una vez.

5. Todos los trabajadores deben trabajar al menos "MinDT" días en el periodo "D".

Se hace un conteo de todos los días que no descansa cada trabajador, teniendo que sumar esta cuenta por lo menos la cantidad de MinDT días.

6. Un trabajador que finaliza en el turno 3 no puede iniciar en el turno 1 del día siguiente.

Se comprueba para todos los días menos el último que si un día trabajas en el último turno implica que no pueda trabajar en el primero el día siguiente.

7. Un trabajador que hace el turno 3 dos días seguidos debe descansar el siguiente día.

De la misma manera que la restricción anterior, todos los días menos los 2 últimos, nos aseguramos de que si trabajas dos noches seguidas eso implica que el siguiente día descanse.

8. Cada trabajador debe estar con al menos "A" trabajadores afines en su turno.

Para cada día y cada trabajador que no descanse, se hace un conteo de todos los trabajadores que trabajan en el mismo turno que él y son compatibles a él, asegurando que dicha cuenta sea igual o superior a la cantidad A establecida.

9. Cada turno debe contar con al menos un trabajador del grupo de encargados "R".

Para turno de cada día se hace una conteo de aquellos trabajadores que son encargados y trabajan en el turno que se está mirando, comprobando que hay al menos un encargado en dicho turno.

2.2. Restricciones de Optimización

Para esta parte de la práctica se tuvieron que editar la estructura de la solución a un array ceros y unos para decir si descansaba o trabajaba en algún turno determinado un trabajador en un día concreto. De tal manera la solución sería una matriz tridimensional donde las filas serían los D días, las columnas los T trabajadores y la profundidad las 4 opciones que tiene un trabajador en un día concreto (0 librar, 1 trabajar 1º turno, 2 trabajar 2º turno y 3 trabajar 3º turno). Pero con la condición especial de que si un trabajador tiene los 4 valores a 0 implica que ese día descansa pero por un motivo especial, que puede ser haber doblado turno el día anterior o haber solicitado el día de descanso, restricciones que se implementan en los apartados 2 y 3.

Para esta versión se incluyen las siguientes restricciones:

1. Implementación de "turnos antiestrés".

De forma muy sencilla se implementa esta restricción adicional, donde si un trabajador trabaja dos mañanas, dos tardes y una noche consecutivas librará dos días seguidos. Esto se lleva a cabo con una ventana de 7 días, donde si se cumplen las condiciones iniciales eso implica que el trabajador descansa dos días seguidos. A estos "turnos antiestrés" solo tienen acceso algunos trabajadores escogidos a dedo que se encuentran en un set recogidos en el archivo de entrada .dzn.

2. Posibilidad de doblar turnos.

Para añadir esta nueva posibilidad primero se determinó un set de trabajadores que tendrían la posibilidad de doblar turnos, es decir, hacer dos turnos seguidos un mismo día para descansar el día siguiente. Lo siguiente fue cambiar la restricción básica 2 que solo permitía tener un turno en un día a cada empleado, permitiendo a los trabajadores que tengan esta posibilidad tener como mucho 2 turnos. A continuación, se añadió la restricción de que si lo anterior sucedía el día siguiente tenía todos los valores a 0 y viceversa; y la restricción de que si un trabajador con opción a doblar tiene 2 turnos un día son dos consecutivos y no son compatibles con la opción de descansar dicho día.

Al añadir esta posibilidad, podemos reducir el número de MaxDT a 4 en el Ejemplo.dzn inicial y saldría satisfactible, algo que no sucedía sin esta nueva funcionalidad.

3. Incorporación de peticiones de días libres y optimización minimizando incumplimientos de forma distribuida.

Para ello hemos creado un array donde los trabajadores puedan indicar qué días no quieren trabajar. Después, minimizamos los días que no se les conceden mediante la suma de todos los días y todos los trabajadores que han pedido el día y no se lo han concedido.

Para repartir equitativamente el cumplimiento de los días solicitados, pensamos en darle más peso a los trabajadores que más días no se le hayan concedido, con lo que se nos ocurrió utilizar el sumatorio sobre los días no concedidos a un trabajador para darle más peso a los trabajadores que más días no les hayan concedido, y así al minimizarlo, el solver buscará que cada trabajador tenga equitativamente el menor número de incumplimientos.

4. Consideración de turnos no deseados y minimización de incumplimientos de forma distribuida.

Para ello hemos creado un array donde los trabajadores puedan indicar qué turnos no quieren trabajar, ya sea un turno concreto todos los días o un turno distinto cada día. Después, minimizamos los turnos que no se les conceden mediante la suma de todos los turnos de todos los días y todos los trabajadores que han pedido el turno libre y no se lo han concedido.

Para repartir equitativamente el cumplimiento de los turnos solicitados, se hace exactamente igual que en apartado anterior.

NOTA: Para los turnos no deseados, se pueden poner tanto en formato único, un turno que no quiere el trabajador ningún día, como diario, que turno no quiere el trabajador cada día. Estas dos formas, son compatibles, pero para asegurar su correcto funcionamiento, si un trabajador no quiere núnca turno de tarde y luego el día dos no quiere ni ni mañana ni tarde, se pondrá en el array de turnos únicos el turno de tarde, y en array de turno para cada día, en el día dos solo el turno de mañana, ya el turno de tarde lo considerará la primera parte.

2.3. Restricciones extras

Hemos añadido una optimización nueva que es repartir los turnos que se trabajan entre todos los días, tratando que todos los trabajadores trabajen la misma cantidad de cada turno. Esto lo hemos hecho penalizando al igual que en el apartado 3 y 4.

Para darle más emoción, hemos optimizado tanto el apartado 3 como el 4 juntos para que la repartición de turnos sea lo más completa posible. Como consideramos más importante que turnos no quieres a que días no quieres, le hemos dado más peso a la optimización de los días.

5. Juegos de Prueba

5.1. Asserts

- Asserts1.dzn con satisfacción.mzn:

```
Running satisfaccion.mzn, Asserts1.dzn
satisfaccion:39.12-88
  in call 'assert'
MiniZinc: assertion failed: Tiene que haber al menos un día para que se pueda planificar
Process finished with non-zero exit code 1.
Finished in 174msec.
```

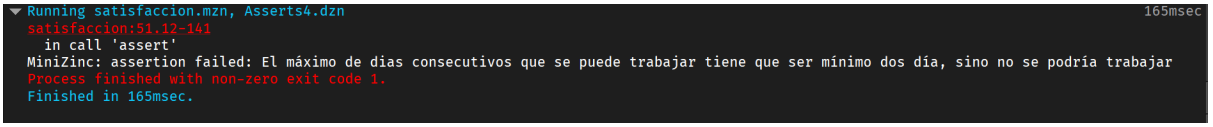
- Asserts2.dzn con satisfacción.mzn:

```
Running satisfaccion.mzn, Asserts2.dzn                                    158msec
satisfaccion:42.12-147
  in call 'assert'
MiniZinc: assertion failed: Tiene que haber al menos tantos trabajadores como trabajadores sean necesarios para cubrir los 3 turnos de un
día
Process finished with non-zero exit code 1.
Finished in 158msec.
```

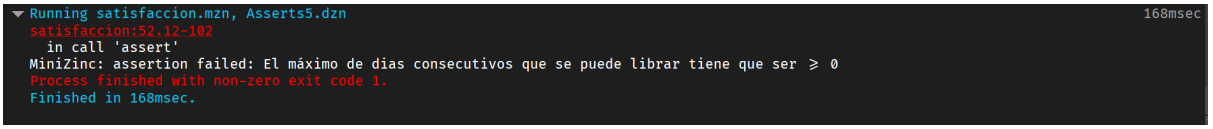
- Asserts3.dzn con satisfacción.mzn:

```
Running satisfaccion.mzn, Asserts3.dzn                                    173msec
satisfaccion:49.12-81
  in call 'assert'
MiniZinc: assertion failed: Tiene que haber al menos un trabajador por turno
Process finished with non-zero exit code 1.
Finished in 173msec.
```

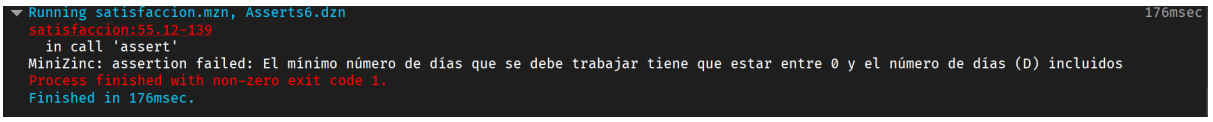
- Asserts4.dzn con satisfacción.mzn:



- Asserts5.dzn con satisfacción.mzn:

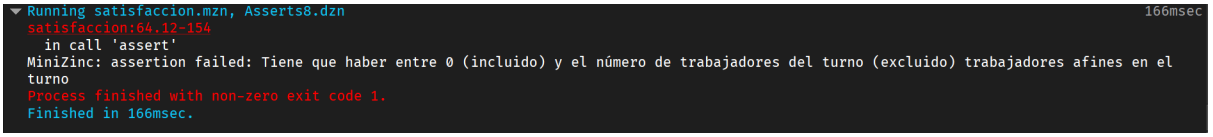


- Asserts6.dzn con satisfacción.mzn:

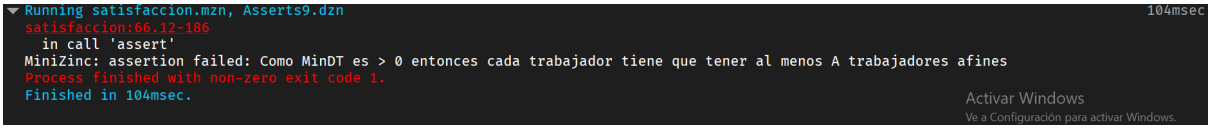


- Asserts7.dzn con satisfacción.mzn:

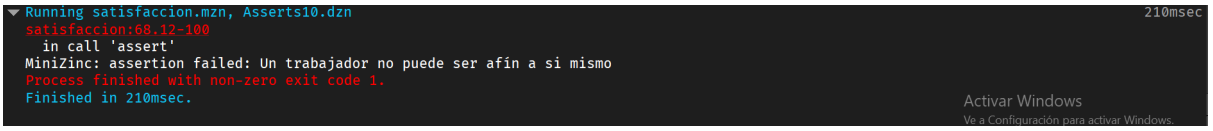
- Asserts8.dzn con satisfacción.mzn:



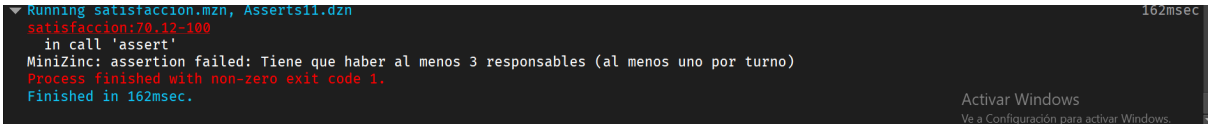
- Asserts9.dzn con satisfacción.mzn:



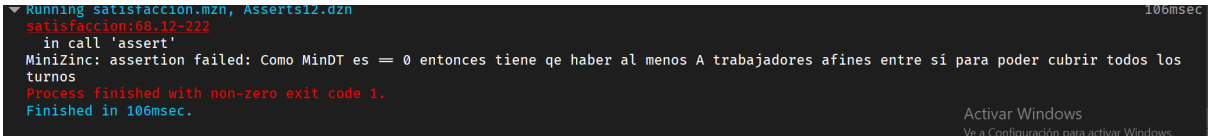
- Asserts10.dzn con satisfacción.mzn:



- Asserts11.dzn con satisfacción.mzn:



- Asserts12.dzn con satisfacción.mzn:



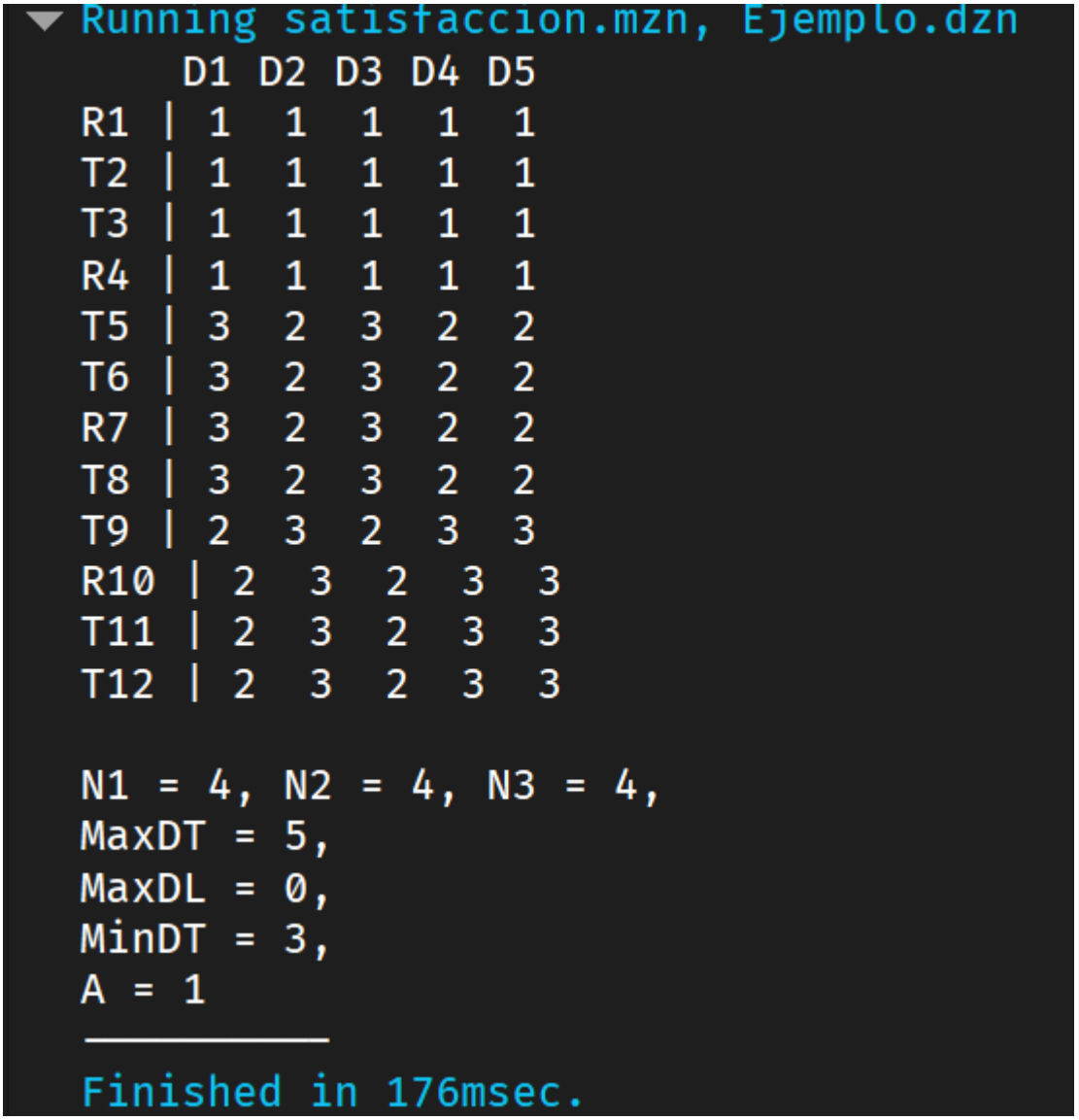
Los asserts son iguales para ambas implementaciones por lo que no hace falta probarlo con optimización msn.

5.2. Restricciones

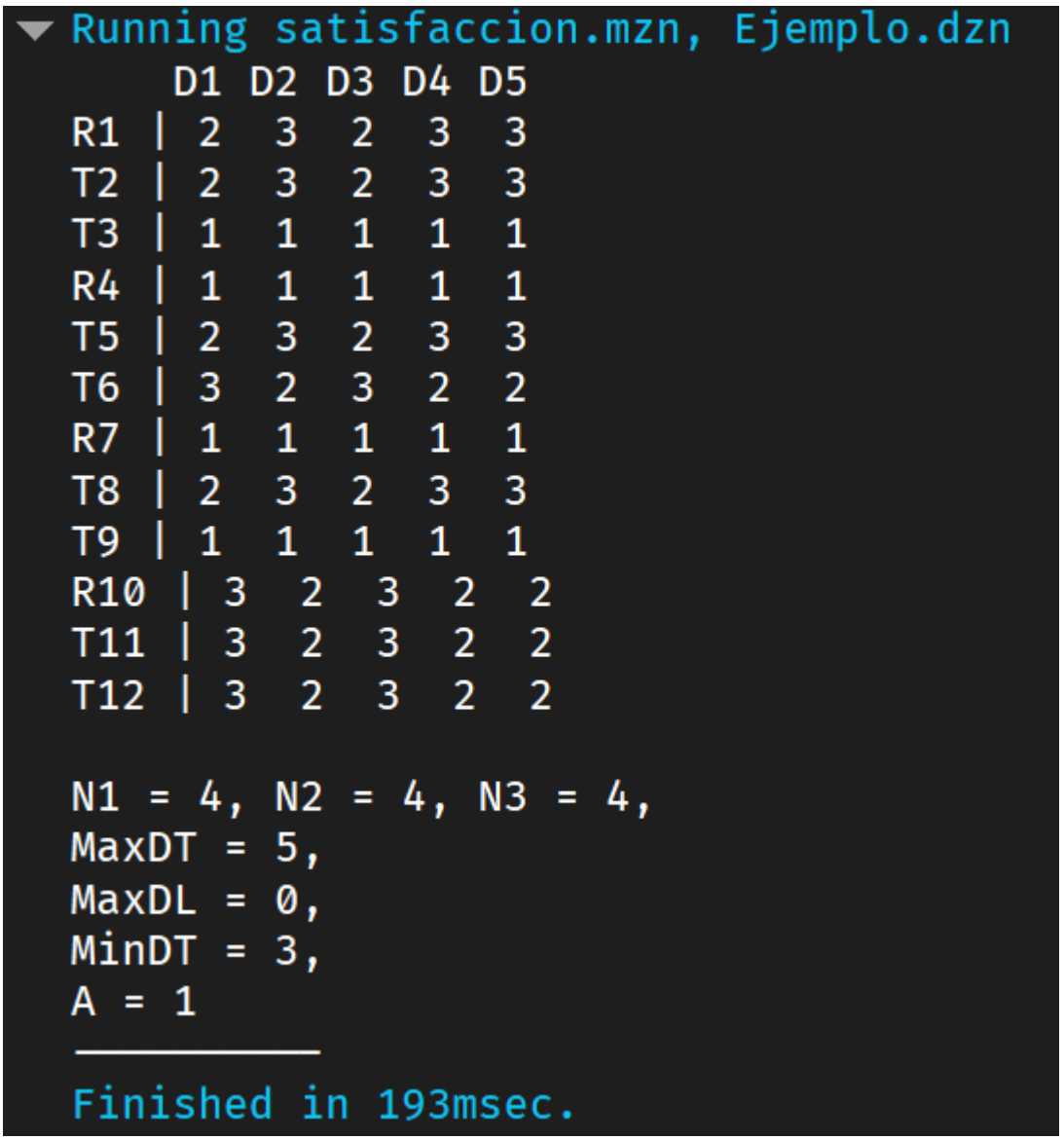
- Ningún trabajador tiene más de "MaxDL" días libres consecutivos y dado un número “MaxDT”, garantizar que nadie trabaja “MaxDT” días consecutivos.

Se ha probado el archivo ejemplo.dzn poniendo MaxDL a 0 y MaxDT a 5 (número de días totales).

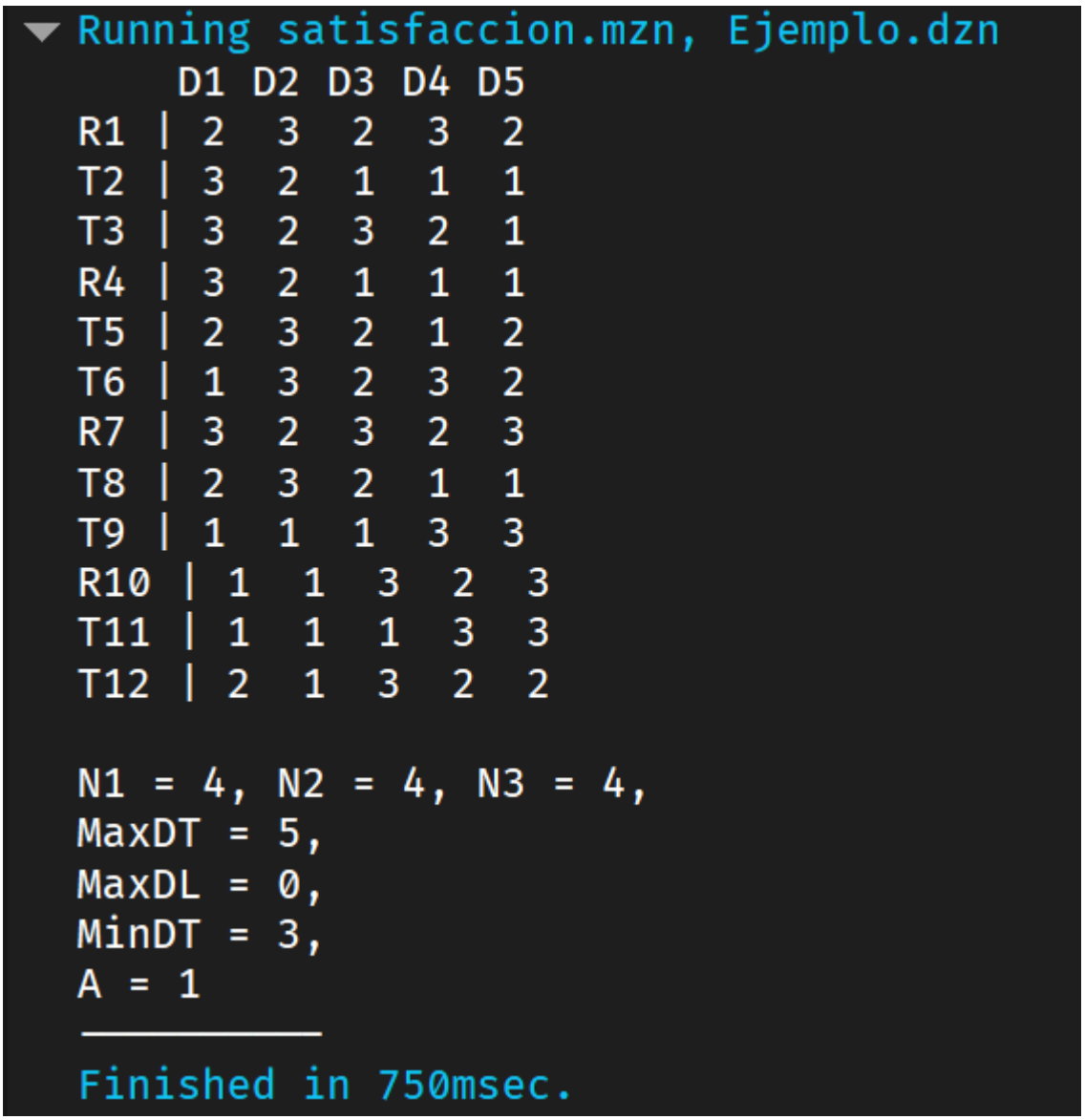
- Satisfaccion.mzn con Chuffed:



- Satisfaccion.mzn con Gecode:

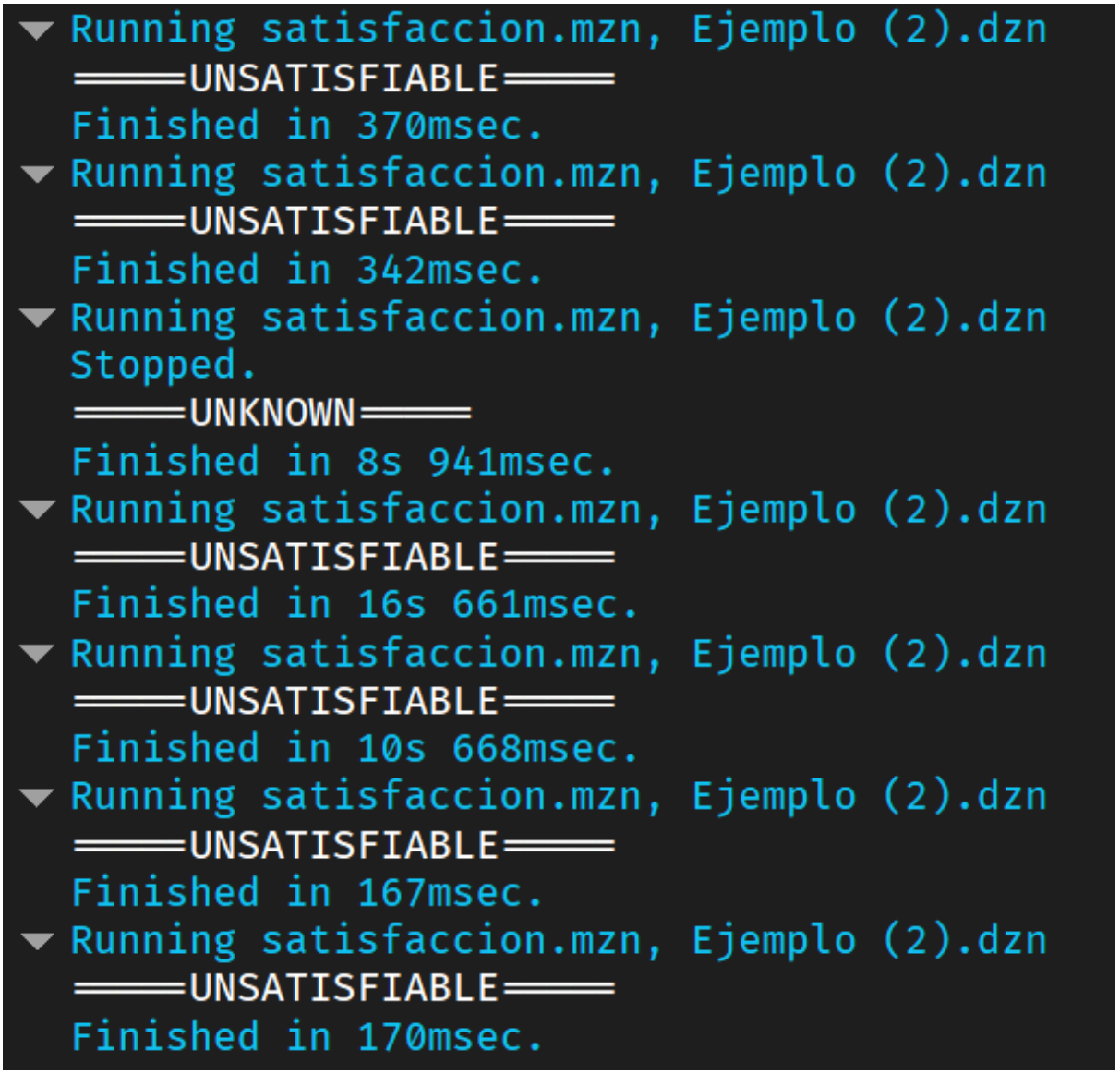


- Satisfaccion.mzn con HiGHS:



- Dado un número "MaxDT", garantizar que nadie trabaja "MaxDT" días consecutivos.

Se ha probado el archivo ejemplo (2).dzn variando en numerosas ocasiones el número MaxDT tratando de reducirlo, cambiando a su vez el valor de MinDT a 0 o 1 y variando también el número de días totales pero no se ha encontrado ninguna solución que satisfaga esta entrada de datos.



- Dado un número “MaxDL”, garantizar que nadie tiene “MaxDL” días libres consecutivos y dado un número “MinDT”, garantizar que todos trabajan como mínimo “MinDT” en los “D” días..

Se ha probado el archivo ejemplo (3).dzn poniendo MaxDL a 5, MaxDT a 5 y MinDT a 0, reduciendo también el número de turnos totales a 6 (2, 2 y 2). Para comprobar que pueden estar los 5 días sin trabajar sin problema porque no exceden el rango de días especificado y además cumplen la condición de que no tienen porqué trabajar puesto que MinDT es 0.

- Satisfaccion.mzn con Chuffed:

```
Running satisfaccion.mzn, Ejemplo (3).dzn
      D1 D2 D3 D4 D5
  R1 |  2  1  0  3  0
  T2 |  0  0  1  1  0
  T3 |  0  1  0  0  0
  R4 |  3  2  1  2  2
  T5 |  2  0  0  3  2
  T6 |  1  0  2  0  0
  R7 |  1  3  2  1  1
  T8 |  0  0  0  0  0
  R9 |  0  3  3  0  3
  T10 | 0  0  0  0  1
  T11 | 3  2  0  2  0
  T12 | 0  0  3  0  3

  N1 = 2, N2 = 2, N3 = 2,
  MaxDT = 5,
  MaxDL = 5,
  MinDT = 0,
  A = 1
  ----------
  Finished in 160msec.
```

- Satisfaccion.mzn con Gecode:

```
Running satisfaccion.mzn, Ejemplo (3).dzn
      D1 D2 D3 D4 D5
 R1 |  2  3  2  3  3
 T2 |  1  2  3  2  0
 T3 |  0  0  0  0  3
 R4 |  3  2  3  2  2
 T5 |  2  3  2  3  0
 T6 |  0  0  0  0  0
 R7 |  1  1  1  1  1
 T8 |  0  0  0  0  0
 R9 |  0  0  0  0  0
 T10 |  0  1  1  1  1
 T11 |  3  0  0  0  2
 T12 |  0  0  0  0  0

 N1 = 2, N2 = 2, N3 = 2,
 MaxDT = 5,
 MaxDL = 5,
 MinDT = 0,
 A = 1
 ----------
 Finished in 151msec.
```

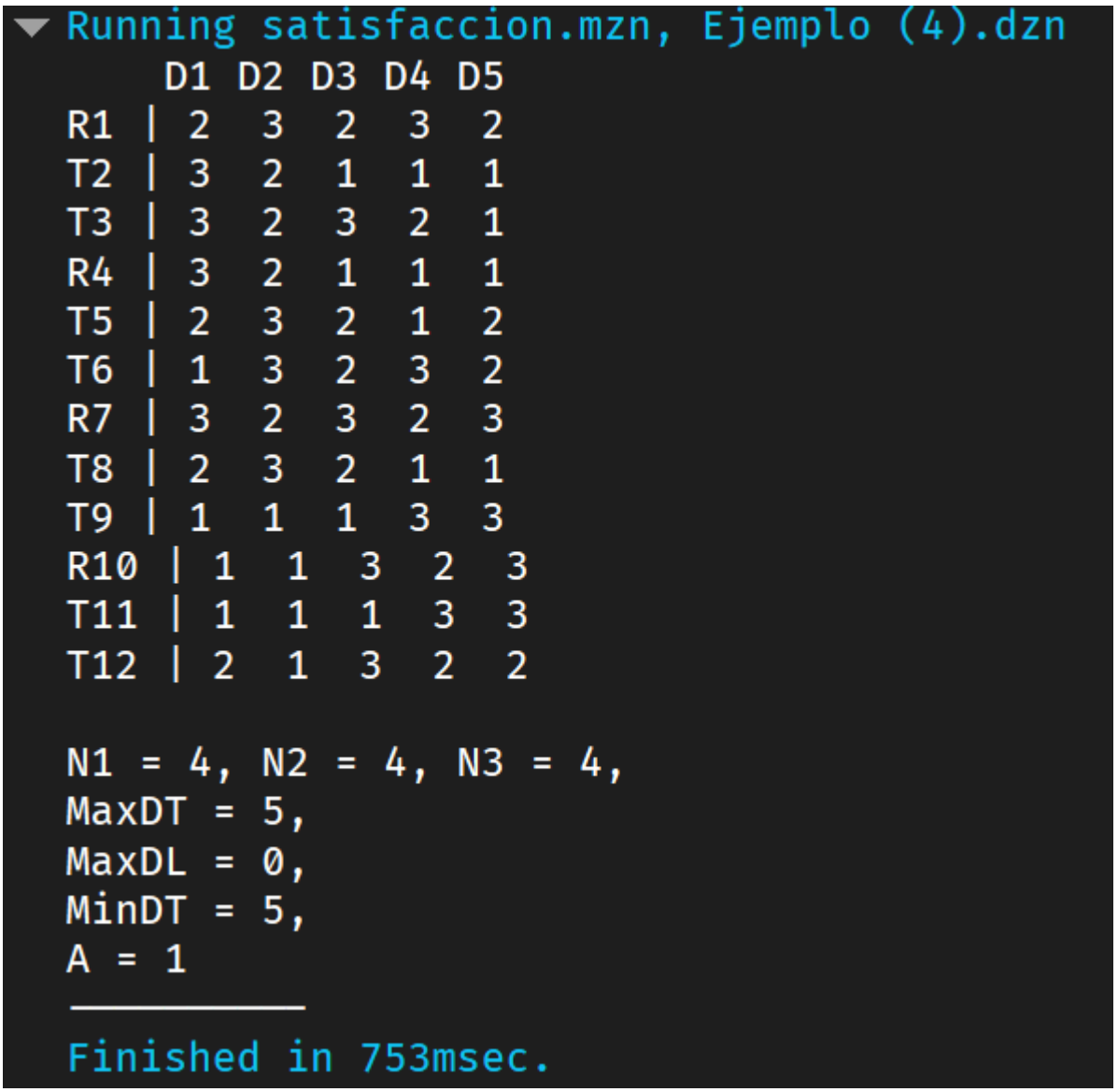
- Satisfaccion.mzn con HiGHS:

```
Running satisfaccion.mzn, Ejemplo (3).dzn
      D1 D2 D3 D4 D5
 R1 |  0  0  2  1  1
 T2 |  0  0  0  0  0
 T3 |  0  0  2  0  1
 R4 |  3  2  1  3  0
 T5 |  0  0  1  1  0
 T6 |  1  3  0  0  3
 R7 |  1  3  0  2  3
 T8 |  0  0  0  0  0
 R9 |  2  1  3  0  2
 T10 | 2  1  3  2  0
 T11 | 3  2  0  3  0
 T12 | 0  0  0  0  2

 N1 = 2, N2 = 2, N3 = 2,
 MaxDT = 5,
 MaxDL = 5,
 MinDT = 0,
 A = 1
 ----------
 Finished in 774msec.
```

- Dado un número "MinDT", garantizar que todos trabajan como mínimo "MinDT" en los "D" días.

Se ha probado el archivo ejemplo (4).dzn poniendo MinDT a 5 para verificar que todos los días trabajan todos los trabajadores.

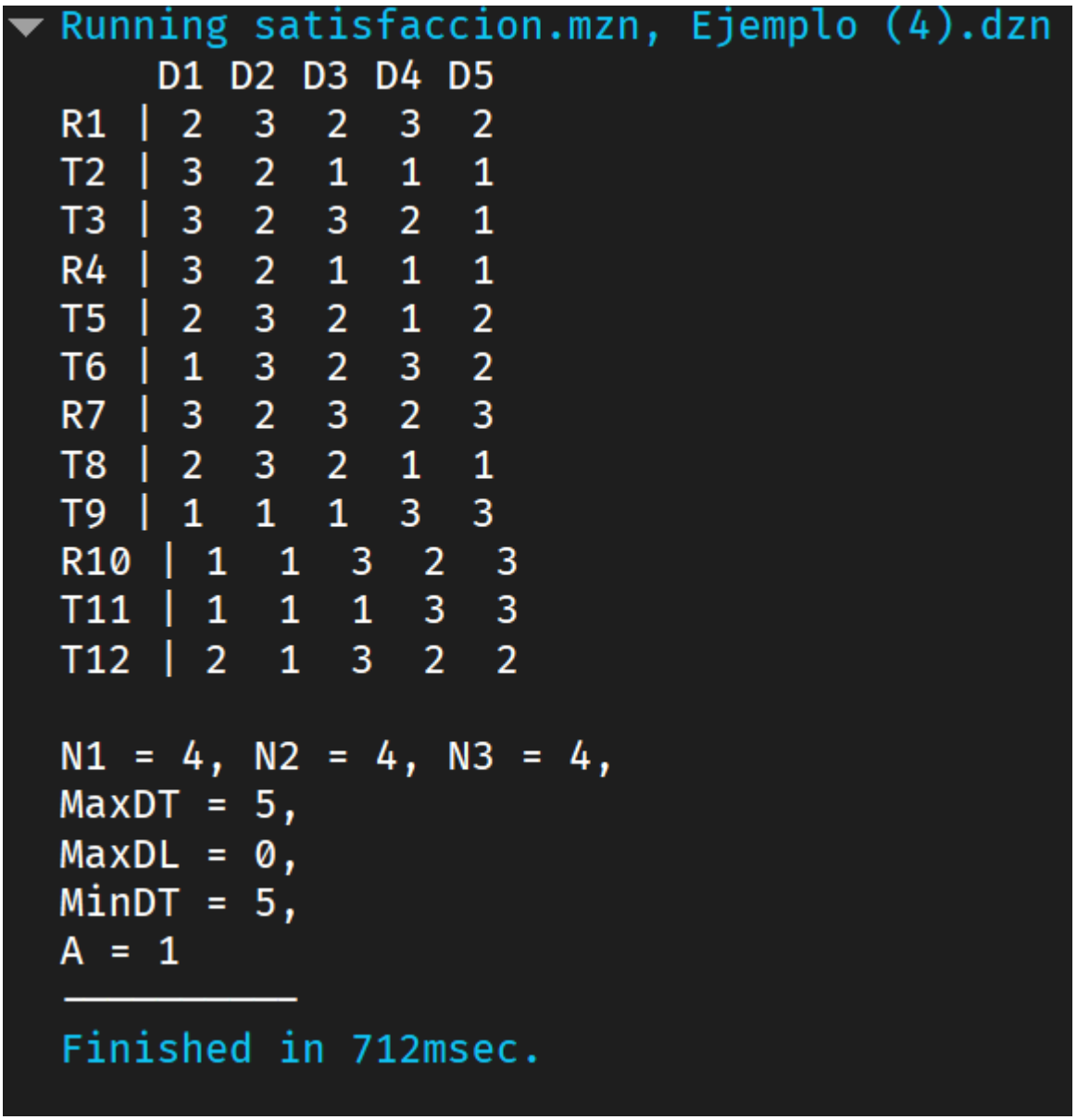
- Satisfaccion.mzn con Chuffed:



- Satisfaccion.mzn con Gecode:

```
Running satisfaccion.mzn, Ejemplo (4).dzn
      D1 D2 D3 D4 D5
 R1 |  2  3  2  3  3
 T2 |  2  3  2  3  3
 T3 |  1  1  1  1  1
 R4 |  1  1  1  1  1
 T5 |  2  3  2  3  3
 T6 |  3  2  3  2  2
 R7 |  1  1  1  1  1
 T8 |  2  3  2  3  3
 T9 |  1  1  1  1  1
 R10 |  3  2  3  2  2
 T11 |  3  2  3  2  2
 T12 |  3  2  3  2  2

 N1 = 4, N2 = 4, N3 = 4,
 MaxDT = 5,
 MaxDL = 0,
 MinDT = 5,
 A = 1
 ----------
 Finished in 192msec.
```

- Satisfaccion.mzn con HiGHS:



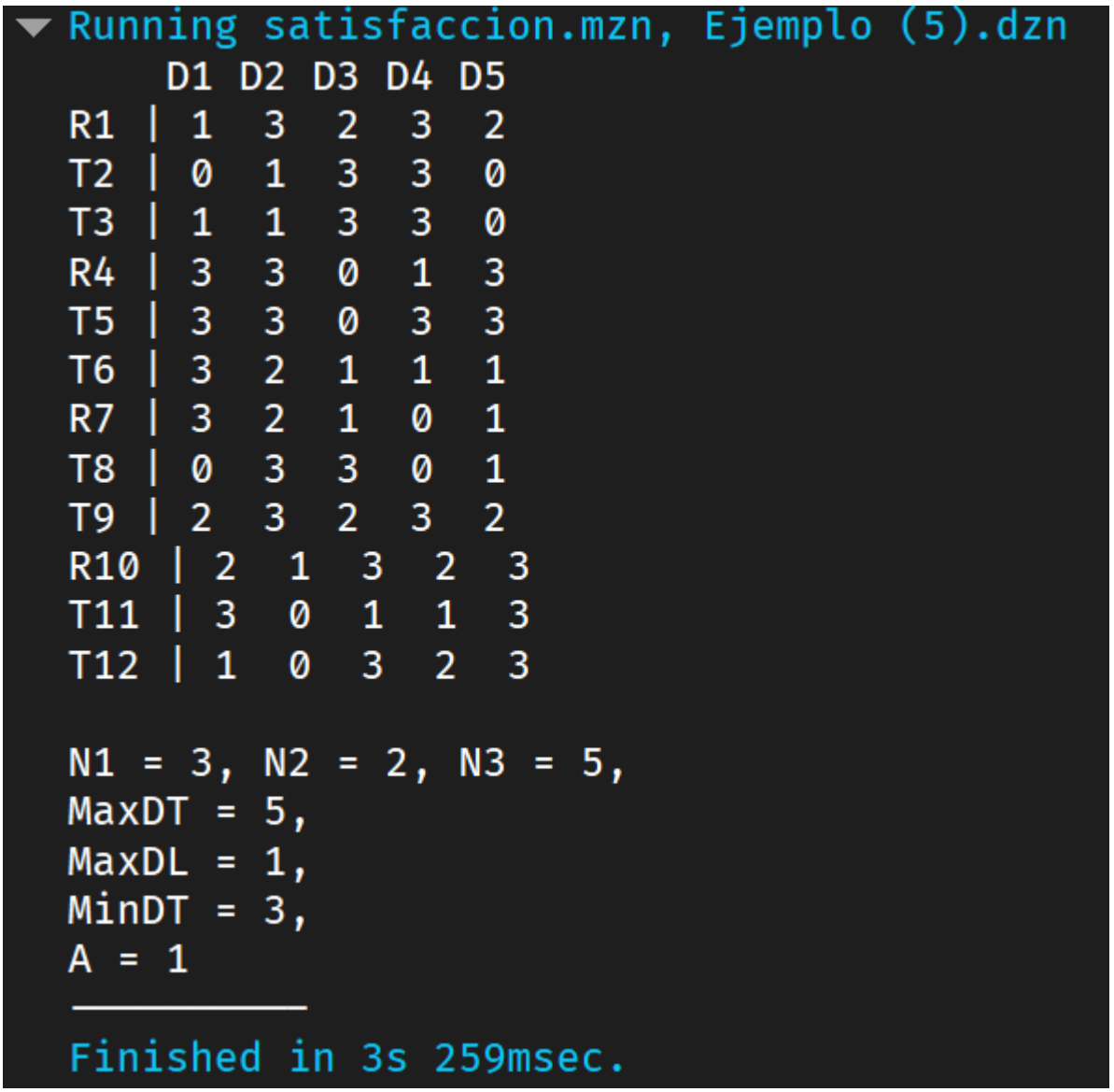
- Si un trabajador hace el último turno de un día entonces no puede tener el primero del día siguiente y si un trabajador hace el último turno dos días seguidos entonces tiene que librar el día siguiente.

Estas restricciones se pueden apreciar en todos los ejemplos anteriores, aunque ejecutando el ejemplo (5).dzn se ha aumentado el número de turnos de noche y de mañana para que haya más turnos de noche y así poder apreciar bien cuando se descansa si has hecho doble turno o no te toca el primer turno si has hecho el último.

- Satisfaccion.mzn con Chuffed:

```
Running satisfaccion.mzn, Ejemplo (5).dzn
      D1 D2 D3 D4 D5
  R1 |  1  3  3  0  2
  T2 |  3  3  0  1  1
  T3 |  1  0  3  3  0
  R4 |  3  2  1  1  1
  T5 |  3  3  0  1  2
  T6 |  0  1  1  2  1
  R7 |  2  1  3  2  3
  T8 |  0  1  3  3  0
  T9 |  2  3  3  0  3
 R10 |  1  3  2  3  3
 T11 |  3  2  1  3  3
 T12 |  3  0  2  3  3

 N1 = 3, N2 = 2, N3 = 5,
 MaxDT = 5,
 MaxDL = 1,
 MinDT = 3,
 A = 1
 ----------
 Finished in 169msec.
```

- Satisfaccion.mzn con Gecode:



- Satisfaccion.mzn con HiGHS:

```
Running satisfaccion.mzn, Ejemplo (5).dzn
      D1 D2 D3 D4 D5
 R1 |  3  2  3  2  2
 T2 |  1  1  1  1  1
 T3 |  0  1  1  2  3
 R4 |  2  3  2  1  1
 T5 |  3  2  3  3  0
 T6 |  3  0  3  0  3
 R7 |  1  3  2  3  3
 T8 |  3  0  3  3  0
 T9 |  3  3  0  3  2
R10 |  1  1  1  3  3
T11 |  2  3  0  1  1
T12 |  0  3  3  0  3

N1 = 3, N2 = 2, N3 = 5,
MaxDT = 5,
MaxDL = 1,
MinDT = 3,
A = 1
----------
Finished in 1s 57msec.
```

- Dada una serie de parejas de trabajadores afines, que se indicarán con una matriz 1..T x 1..T de Booleanos “afines”, y un número “A”, cada trabajador de un turno tiene que estar con al menos A trabajadores afines en ese turno.

Se ha probado el archivo ejemplo (6).dzn donde todos los trabajadores son afines entre sí para ver que no hay ningún problema con dicha restricción. Nos gustaría encontrar el caso de prueba contrario donde cada empleado pueda trabajar específicamente con un único compañero, pero sería como ir colocando a dedo al trabajador y haría siempre el turno con el mismo compañero.

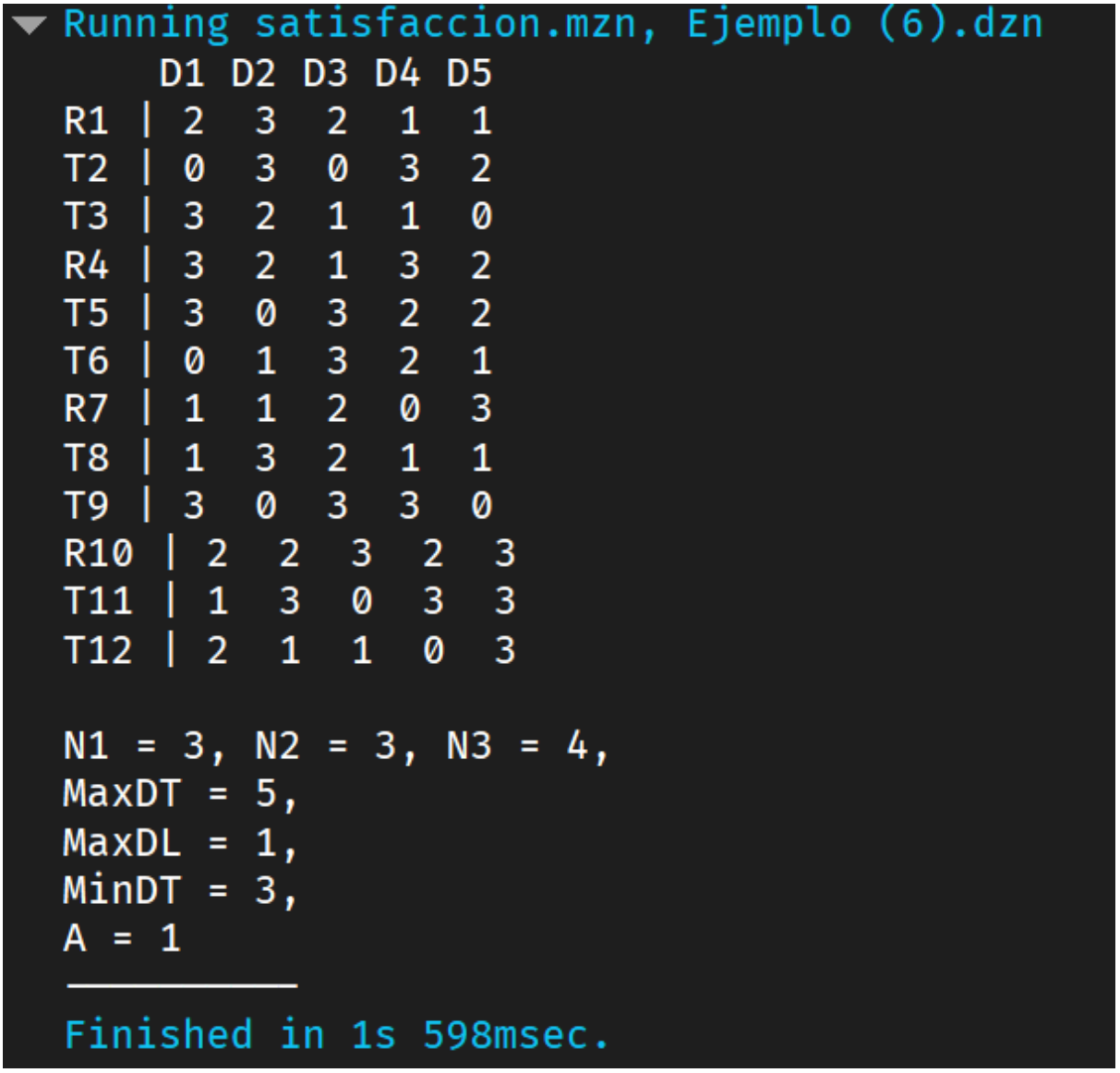
- Satisfaccion.mzn con Chuffed:

```
Running satisfaccion.mzn, Ejemplo (6).dzn
     D1 D2 D3 D4 D5
R1 | 1  0  1  1  0
T2 | 1  1  0  1  0
T3 | 3  0  3  0  1
R4 | 0  1  1  0  1
T5 | 0  3  0  1  1
T6 | 1  1  3  2  2
R7 | 3  2  3  2  2
T8 | 3  2  3  2  2
T9 | 3  2  1  3  3
R10 | 2  3  2  3  3
T11 | 2  3  2  3  3
T12 | 2  3  2  3  3

N1 = 3, N2 = 3, N3 = 4,
MaxDT = 5,
MaxDL = 1,
MinDT = 3,
A = 1
----------
Finished in 181msec.
```

- Satisfaccion.mzn con Gecode:

```
Running satisfaccion.mzn, Ejemplo (6).dzn
     D1 D2 D3 D4 D5
R1  | 3  2  3  2  2
T2  | 3  2  3  3  0
T3  | 3  2  3  3  0
R4  | 0  1  0  1  1
T5  | 1  3  2  3  3
T6  | 2  1  1  1  3
R7  | 1  0  1  0  1
T8  | 3  3  0  1  3
T9  | 1  0  1  0  1
R10 | 2  3  2  3  3
T11 | 0  3  2  2  2
T12 | 2  1  3  2  2

N1 = 3, N2 = 3, N3 = 4,
MaxDT = 5,
MaxDL = 1,
MinDT = 3,
A = 1
----------
Finished in 192msec.
```

- Satisfaccion.mzn con HiGHS:



- Sea “R” el conjunto de trabajadores (que se obtendrá como un set de números de trabajador) que tienen la categoría de encargados. En cada turno debe haber al menos un responsable.

Se ha probado el archivo ejemplo (7).dzn poniendo donde el número de encargados coincide con el número de turnos siendo el 1, 4 y 7; donde se aprecia que los 3 trabajan todos los días para poder satisfacer esta condición.

- Satisfaccion.mzn con Chuffed:

```
Running satisfaccion.mzn, Ejemplo (7).dzn
      D1 D2 D3 D4 D5
 R1 |  3  2  1  3  2
 T2 |  3  0  3  0  1
 T3 |  3  0  1  3  0
 R4 |  1  1  3  2  1
 T5 |  0  1  3  2  2
 T6 |  1  3  3  0  3
 R7 |  2  3  2  1  3
 T8 |  0  1  0  2  2
 T9 |  2  2  2  1  1
 T10 |  3  3  0  1  0
 T11 |  1  2  1  3  3
 T12 |  2  3  2  3  3

 N1 = 3, N2 = 3, N3 = 4,
 MaxDT = 5,
 MaxDL = 1,
 MinDT = 3,
 A = 1
 ----------
 Finished in 174msec.
```

- Satisfaccion.mzn con Gecode:

```
Running satisfaccion.mzn, Ejemplo (7).dzn
      D1 D2 D3 D4 D5
 R1 |  2  3  2  3  3
 T2 |  2  3  2  2  3
 T3 |  1  0  1  0  1
 R4 |  3  2  3  2  2
 T5 |  2  3  2  3  3
 T6 |  3  2  0  1  2
 R7 |  1  1  1  1  1
 T8 |  3  3  0  1  3
 T9 |  1  0  1  0  1
 T10 | 0  1  3  3  0
 T11 | 3  2  3  2  2
 T12 | 0  1  3  3  0

 N1 = 3, N2 = 3, N3 = 4,
 MaxDT = 5,
 MaxDL = 1,
 MinDT = 3,
 A = 1
 ----------

 Finished in 161msec.
```

- Satisfaccion.mzn con HiGHS:

```
Running satisfaccion.mzn, Ejemplo (7).dzn
      D1 D2 D3 D4 D5
 R1 |  2  1  3  2  2
 T2 |  1  3  3  0  3
 T3 |  2  1  2  2  2
 R4 |  1  3  2  3  3
 T5 |  3  2  3  3  0
 T6 |  1  0  1  1  3
 R7 |  3  2  1  1  1
 T8 |  3  0  1  3  0
 T9 |  2  2  2  2  2
 T10 | 3  3  0  1  1
 T11 | 0  1  0  3  3
 T12 | 0  3  3  0  1

 N1 = 3, N2 = 3, N3 = 4,
 MaxDT = 5,
 MaxDL = 1,
 MinDT = 3,
 A = 1
 ----------
 Finished in 637msec.
```

pruebas de todos los solvers con el ejemplo básico

```
Running satisfaccion.mzn, Ejemplo.dzn
     D1 D2 D3 D4 D5
R1 | 0  3  2  1  0
T2 | 1  0  1  1  1
T3 | 3  0  1  0  1
R4 | 1  3  3  0  1
T5 | 2  3  0  1  0
T6 | 3  2  3  2  2
R7 | 3  2  3  2  2
T8 | 3  2  3  2  2
T9 | 0  1  2  3  3
R10 | 2  1  1  3  3
T11 | 1  1  0  3  3
T12 | 2  3  2  3  3

N1 = 3, N2 = 3, N3 = 4,
MaxDT = 5,
MaxDL = 1,
MinDT = 3,
A = 1
----------
Finished in 223msec.
```

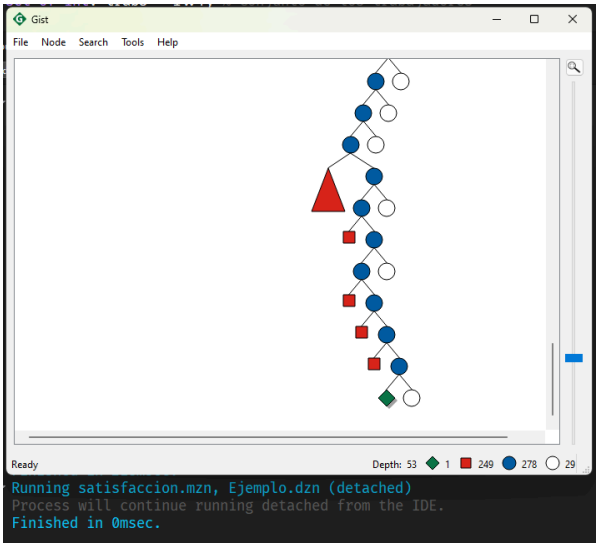
```
Running satisfaccion.mzn, Ejemplo.dzn
     D1 D2 D3 D4 D5
R1 | 3  2  0  1  1
T2 | 2  1  2  1  2
T3 | 1  1  3  2  2
R4 | 2  3  2  3  0
T5 | 3  0  1  1  1
T6 | 0  3  2  0  3
R7 | 2  0  3  2  3
T8 | 1  3  3  0  1
T9 | 3  2  3  2  3
R10 | 1  1  1  3  2
T11 | 3  3  0  3  0
T12 | 0  2  1  3  3

N1 = 3, N2 = 3, N3 = 4,
MaxDT = 5,
MaxDL = 1,
MinDT = 3,
A = 1
----------
Finished in 2s 242msec.
```

chuffed, coin-bc,

```
Running satisfaccion.mzn, Ejemplo.dzn
Error: error: Cannot open file 'C:\Users\Usuario\OneDrive - Universidad Complutense de Madrid (UCM)\Optativas\PR\Prácticas\Práctica 1\satisfaccion.mzn'.
Finished in 108msec.
```

findMus mucho mus y poco find

```
Running satisfaccion.mzn, Ejemplo.dzn
     D1 D2 D3 D4 D5
R1 | 2  3  2  3  2
T2 | 1  1  3  3  0
T3 | 2  3  2  1  0
R4 | 0  1  0  1  1
T5 | 0  1  1  3  2
T6 | 3  2  3  2  3
R7 | 1  0  1  0  1
T8 | 3  3  0  3  2
T9 | 1  0  1  0  1
R10 | 3  2  3  2  3
T11 | 2  3  2  1  3
T12 | 3  2  3  2  3

N1 = 3, N2 = 3, N3 = 4,
MaxDT = 5,
MaxDL = 1,
MinDT = 3,
A = 1
----------
Finished in 218msec.
```



geocode y geocode gist (su interfaz gráfica)

como podemos ver, en el caso básico quien mejor se desenvuelve son chuffed y geocode
Estamos cansados Pablo, no nos da a más

5.1. Optimización

```
Running optimizacion.mzn, optimizacion.dzn
     D1 D2 D3 D4 D5
 R1 | 1000  0010  0100  1000  0110
 T2 | 0010  0001  0001  1000  0010
 T3 | 0001  0010  1000  0100  0001
 R4 | 0100  0001  0010  0110  0000
 T5 | 0011  0000  0100  0001  0010
 T6 | 0100  0100  0010  0001  0001
 R7 | 0010  0100  0001  0001  1000
 T8 | 1000  0100  0100  0001  0001
 T9 | 0001  0001  1000  0010  0100
 R10 | 0001  0010  0001  0010  0001
 T11 | 0100  0000  0010  0100  0100
 T12 | 0000  0001  0001  1000  0000

 N1 = 3, N2 = 3, N3 = 4,
 MaxDT = 5,
 MaxDL = 1,
 MinDT = 3,
 A = 1
 objetivo = 0
 ----------
 ==========
 Finished in 4s 180msec.
```

minimizando el número de veces que el trabajador 1 hace el turno de tarde

```
Running optimizacion.mzn, optimizacion.dzn
    D1 D2 D3 D4 D5
R1 | 1000  0010  0100  0001  1000
T2 | 0001  1000  0001  1000  0110
T3 | 0100  0100  0001  0010  0100
R4 | 0001  0001  1000  0100  0010
T5 | 0001  0010  0100  0001  0010
T6 | 0010  0001  0010  0001  0001
R7 | 0010  0001  0010  0100  0001
T8 | 0010  0010  0100  0001  1000
T9 | 0100  0100  0001  0010  0001
R10 | 0100  0100  0001  0010  0100
T11 | 0001  0000  0000  0100  0001
T12 | 0000  0001  0010  0000  0000

N1 = 3, N2 = 3, N3 = 4,
MaxDT = 5,
MaxDL = 1,
MinDT = 3,
A = 1
objetivo = 3
---------
=========
Finished in 3s 619msec.
```

minimizando el número de veces que el trabajador 1 hace el turno de mañana, tarde y noche

```
Running optimizacion.mzn, optimizacion.dzn
     D1 D2 D3 D4 D5
 R1 | 0001  1000  0001  0001  1000
 T2 | 0011  0000  0100  0010  0010
 T3 | 0001  1000  0001  0001  1000
 R4 | 0100  0011  0000  0100  0011
 T5 | 1000  0001  1000  0011  0000
 T6 | 0100  0100  0011  0000  0100
 R7 | 0010  0100  0100  0010  0100
 T8 | 1000  0001  0001  1000  0100
 T9 | 0001  0010  0100  0100  0010
 R10 | 0010  0100  0010  0001  0001
 T11 | 0100  0001  0000  0100  0001
 T12 | 0000  0010  0010  0000  0001

 N1 = 3, N2 = 3, N3 = 4,
 MaxDT = 5,
 MaxDL = 1,
 MinDT = 3,
 A = 1
 objetivo = 0
 ----------
 ==========
 Finished in 3s 305msec.
```

minimizando el número de veces que el trabajador 1 hace el turno de mañana y tarde

```
Running optimizacion.mzn, optimizacion.dzn
     D1 D2 D3 D4 D5
 R1 | 0001  0001  1000  0100  0010
 T2 | 0011  0000  0100  1000  0001
 T3 | 0001  0010  0100  0100  0100
 R4 | 0100  0001  0010  0001  0001
 T5 | 0100  0001  0010  0001  0001
 T6 | 1000  0100  0001  0001  1000
 R7 | 0010  0100  0100  0010  0010
 T8 | 1000  0001  1000  0001  0001
 T9 | 0010  0010  0001  0010  0010
 R10 | 0001  0010  0001  0010  0100
 T11 | 0100  0100  0010  0100  0100
 T12 | 0000  1000  0001  0000  0000

 N1 = 3, N2 = 3, N3 = 4,
 MaxDT = 5,
 MaxDL = 1,
 MinDT = 3,
 A = 1
 objetivo = 0
 ----------
 ==========
 Finished in 3s 263msec.
```

```
     D1 D2 D3 D4 D5 D6 D7
R1 | 0100  0100  0001  0010  1000  0100  0100
T2 | 0010  0010  0010  0001  0001  1000  0010
T3 | 0100  0100  0001  0010  1000  0100  0010
R4 | 0010  0001  0010  0001  0001  1000  0001
T5 | 0010  0001  1000  0001  0010  0100  0100
T6 | 0001  0010  0100  1000  0100  0010  0001
R7 | 0001  0010  0100  1000  0100  0010  0001
T8 | 0100  0001  1000  0010  0010  0010  0100
R9 | 0001  1000  0100  0100  0010  0001  0010
T10 | 0000  0000  0000  0100  0100  0001  0000
T11 | 0000  0100  0010  0100  0001  0000  0000
T12 | 0000  0000  0001  0000  0000  0001  0000

N1 = 3, N2 = 3, N3 = 3,
MaxDT = 5,
MaxDL = 1,
MinDT = 3,
A = 1
objetivo = 9
 ──────────
 ══════════
Finished in 6s 378msec.
```

sin repartir

```
     D1 D2 D3 D4 D5 D6 D7
R1 | 0010  0100  0010  0100  1000  0001  0010
T2 | 0001  1000  0001  0010  0100  0010  0001
T3 | 0010  0001  0010  0100  1000  0001  0010
R4 | 0100  1000  0100  0010  0100  0010  0001
T5 | 0100  0100  0100  0001  0010  1000  0010
T6 | 0001  0010  0001  1000  0001  0010  0100
R7 | 0001  0010  0001  0010  0001  1000  0100
T8 | 0010  0100  0010  0001  0001  1000  0100
R9 | 0100  0001  1000  0001  0010  0100  0001
T10 | 0000  0010  0000  0000  0010  0100  0000
T11 | 0000  0001  0000  0000  0100  0100  0000
T12 | 0000  0000  0100  0100  0000  0001  0000

N1 = 3, N2 = 3, N3 = 3,
MaxDT = 5,
MaxDL = 1,
MinDT = 3,
A = 1
objetivo = 18
_________
=========
Finished in 56s 636msec.
```

repartió 3 dias a cada uno
minimizando el número de veces que el trabajador 1 hace mañana o tarde el primer y segundo día, tarde o noche el tercer o cuarto día y mañana o noche el último día

Mezclando las optimizaciones 3 y 4 cuando el turno que no deseas es común para todos los días y no para uno en concreto.

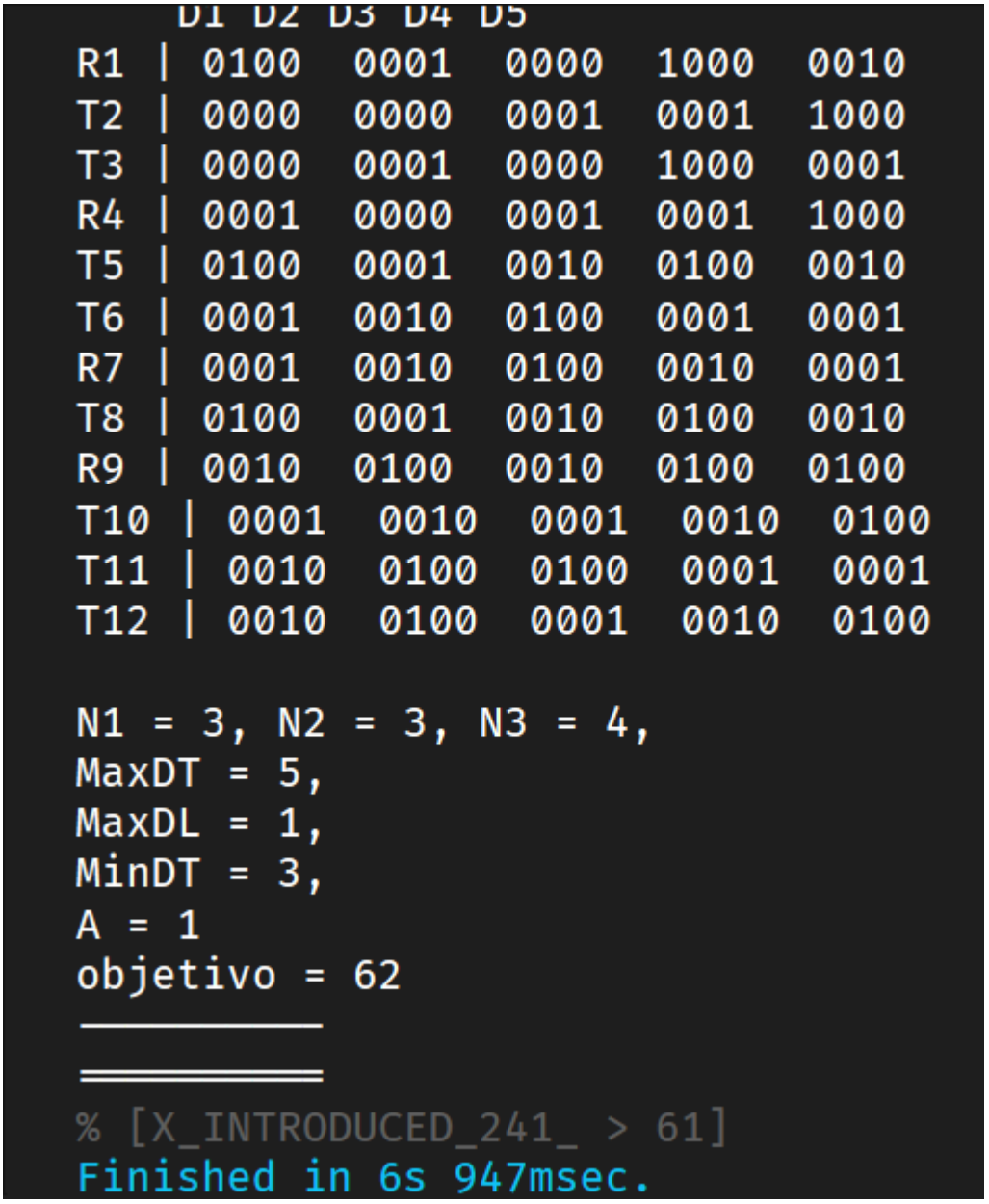
Dada esta entrada de datos:
diasNT = [{1,2,3},{1,2,3},{1,2,3},{1,2,3},{},{},{},{},{},{},{},{}];

turnosNT = [{1,2,3},{1,2,3},{1,2,3},{1,2,3},{},{},{},{},{},{},{},{}];

Se encuentra esta solución con HiGHS:

```
     D1 D2 D3 D4 D5
R1 | 0010  0000  0000  1000  0100
T2 | 0000  0001  0000  0100  1000
T3 | 0001  0000  0001  1000  0100
R4 | 0000  0100  0100  0100  1000
T5 | 0010  0100  0100  0100  0001
T6 | 0100  0001  0010  0001  0010
R7 | 0100  0001  0010  0001  0010
T8 | 0010  0001  0010  0001  0010
R9 | 0001  0010  0001  0010  0001
T10 | 0001  0010  0001  0010  0001
T11 | 0100  0100  0100  0001  0001
T12 | 0001  0010  0001  0010  0100

N1 = 3, N2 = 3, N3 = 4,
MaxDT = 5,
MaxDL = 1,
MinDT = 3,
A = 1
objetivo = 62
----------
==========
Finished in 30s 102msec.
```

Mientras que con Chuffed:



Con el resto de solvers tarda demasiado tiempo como para tenerlo en cuenta al comparar velocidades de resolución.

Mezclando las optimizaciones 3 y 4 cuando el turno que no deseas no es común para todos los días sino que varía según el día concreto.

Dada la entrada de datos:

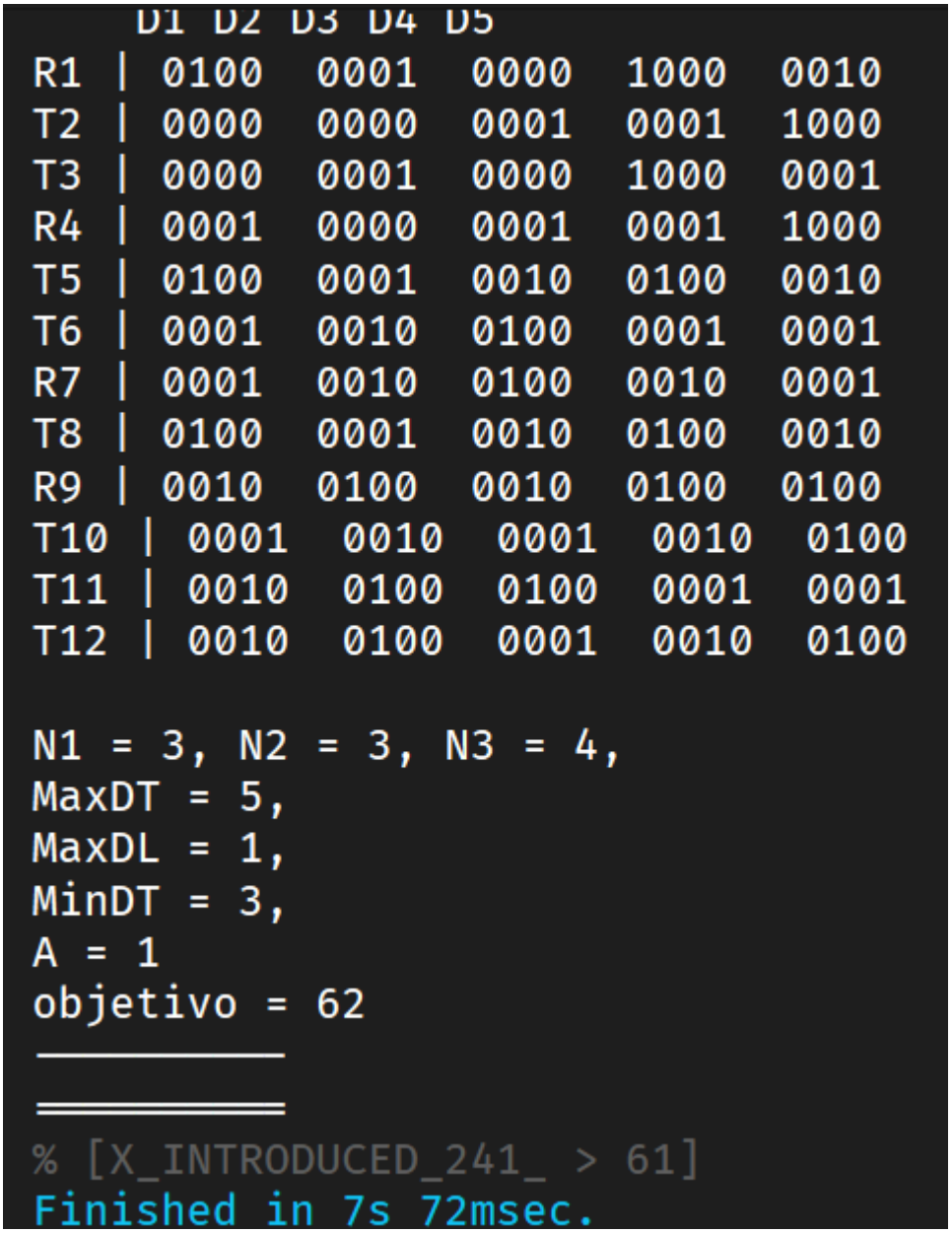
diasNT = [{1,2,3},{1,2,3},{1,2,3},{1,2,3},{},{},{},{},{},{},{},{}];
turnosDiaNT = [|{1,2,3},{1,2,3},{1,2,3},{1,2,3},{},{},{},{},{},{},{},{},
|{1,2,3},{1,2,3},{1,2,3},{1,2,3},{},{},{},{},{},{},{},{},
|{1,2,3},{1,2,3},{1,2,3},{1,2,3},{},{},{},{},{},{},{},{},
|{1,2,3},{1,2,3},{1,2,3},{1,2,3},{},{},{},{},{},{},{},{},

|{1,2,3},{1,2,3},{1,2,3},{1,2,3},{},{},{},{},{},{},{},{}|];

Se encuentra esta solución con HiGHS:

```
     D1 D2 D3 D4 D5
R1 | 0010  0000  0000  1000  0100
T2 | 0000  0001  0000  0100  1000
T3 | 0001  0000  0001  1000  0100
R4 | 0000  0100  0100  0100  1000
T5 | 0010  0100  0100  0100  0001
T6 | 0100  0001  0010  0001  0010
R7 | 0100  0001  0010  0001  0010
T8 | 0010  0001  0010  0001  0010
R9 | 0001  0010  0001  0010  0001
T10 | 0001  0010  0001  0010  0001
T11 | 0100  0100  0100  0001  0001
T12 | 0001  0010  0001  0010  0100

N1 = 3, N2 = 3, N3 = 4,
MaxDT = 5,
MaxDL = 1,
MinDT = 3,
A = 1
objetivo = 62
----------
==========
Finished in 29s 769msec.
```

Mientras que con Chuffed:



Al igual que antes, el resto de solvers dan la solución correcta tras demasiado tiempo como para esperar a conseguirla, por lo que no se les tiene en cuenta siquiera.

La salida de ambas coinciden porque en resumidas cuentas se han usado los mismos datos de entrada pero esto prueba que encuentra la misma solución correcta con ambos métodos, por lo que se puede extrapolar que cambiando los datos también lo haría.

La última optimización adicional es la de tratar de distribuir todos los turnos entre todos los trabajadores de forma igual, para evitar desigualdades y que todos trabajen en las mismas condiciones.

```
    D1 D2 D3 D4 D5
R1 | 0010  0010  0100  0001  0001
T2 | 0001  0010  0100  0001  1000
T3 | 0010  0010  0100  0001  0001
R4 | 0001  1000  0001  0010  0100
T5 | 0001  1000  0001  0010  0100
T6 | 0100  0100  0010  0001  1000
R7 | 0100  0100  0001  1000  0010
T8 | 0100  0100  0001  0010  0001
R9 | 1000  0001  0010  0100  0010
T10 | 0010  0001  1000  0100  0001
T11 | 1000  0001  0010  1000  0100
T12 | 0001  0001  1000  0100  0010

N1 = 3, N2 = 3, N3 = 4,
MaxDT = 5,
MaxDL = 1,
MinDT = 3,
A = 1
objetivo = 64
```

Tras estar 18 minutos ejecutándose y 14 con el mismo objetivo, hemos asumido que esta era la mejor solución posible donde se puede apreciar que en los 5 días como mucho se puede hacer un turno 2 veces. Además el ordenador comenzaba a sonar demasiado por lo que se decidió por detener la ejecución con Chuffed.