Práctica 2: Uso de MiniZinc para resolver problemas sobre la satisfacción de restricciones

UNIVERSIDAD DE GRANADA

Técnicas de los Sistemas Inteligentes

Elena Torres Fernández
Curso académico 2024-2025

11 abril 2025

Índice

1. Problema del cambio de monedas

1.1. Resolución con codificación entera

Importe	Primera solución encontrada y número de monedas en la misma	Número total de soluciones	Runtime (ms)
0,13 Unid.	monedas: 13 0 0 0 0 0 0 0 0 total monedas: 13	12	159
1,47 Unid.	monedas: 147 0 0 0 0 0 0 0 0 total monedas: 147	11555	2258
2,30 Unid.	monedas: 230 0 0 0 0 0 0 0 0 total monedas: 230	77206	13781
2,99 Unid.	monedas: 299 0 0 0 0 0 0 0 0 total monedas: 299	258664	57035

Cuadro 1: Resultados del Problema de la moneda 1a

1.2. Resolución con codificación entera imponiendo que la parte entera de los importes sea asignada únicamente a monedas de una o dos Unidades

Importe	Primera solución encontrada y número de monedas en la misma	Número total de soluciones	Runtime (ms)
0,13 Unid.	monedas: 13 0 0 0 0 0 0 0 0 total monedas: 13	12	164
1,47 Unid.	monedas: 47 0 0 0 0 0 1 0 0 total monedas: 48	230	366
2,30 Unid.	monedas: 30 0 0 0 0 0 0 1 0 total monedas: 31	142	226
2,99 Unid.	monedas: 99 0 0 0 0 0 2 0 0 total monedas: 101	5146	1109

Cuadro 2: Resultados del Problema de la moneda 1b

1.3. Modificación del apartado anterior para que sea un problema de optimización (minimizando el número de monedas)

Importe	Solución óptima	Número de monedas de la solución óptima	Runtime (ms)
0,13 Unid.	monedas: 0 1 0 1 0 0 0 0 0	2	179
1,47 Unid.	monedas: 2 0 0 2 1 0 1 0 0	6	179
2,30 Unid.	monedas: 0 0 1 0 1 0 0 1 0	3	178
2,99 Unid.	monedas: 1 1 0 2 1 1 0 1 0	7	180

Cuadro 3: Resultados del Problema de la moneda 1c

1.4. Preguntas para razonar

1. A medida que vamos aumentando el importe, ¿qué ocurre? ¿Cómo escala (a nivel de tiempos de resolución) nuestra codificación con importes progresivamente más elevados? Por ejemplo, ¿cuánto se tardaría en encontrar todas las soluciones para un importe de 3.5 Unid. y 5.27 Unid.?

   Pruebas con los **nuevos importes**: para 3,5 Unid. tarda 124.000 ms en encontrar las 555.724 soluciones que obtengo; para 5,27 Unid., 17 minutos en encontrar 4.585.725 soluciones. A medida que aumentamos el importe, el runtime crece de forma **exponencial**, pues las soluciones que obtenemos también crecen de forma exponencial al tratarse de todas las posibles combinaciones de monedas, sin ninguna optimización ni restricción adicional. En los siguientes apartados, reducimos drásticamente el tiempo de cómputo porque sí establecemos restricciones, como usar las monedas de unidades para completar la parte entera siempre que sea posible.

2. En relación con el punto anterior, ¿qué ocurriría si, usando la codificación anterior para encontrar todas las soluciones, el importe buscado es mucho mayor (del orden de millones de Unidades)?

   En ese caso estaríamos hablando de unas $9^{1000000000}$ **posibles soluciones**, siendo 9 el número de monedas diferentes que tenemos y el exponente, los céntimos que tendría el importe de entrada. Se trata de un número extremadamente grande que **no es almacenable** en ningún ordenador y mucho menos operable con él.

3. En caso de que haya algún problema con lo planteado en el punto anterior, ¿cuál podría ser una estrategia prometedora para resolverlo?

   Una estrategia prometedora es imponer usar monedas de unidades siempre que sea posible. Con esto estamos reduciendo el espacio de búsqueda de soluciones. Primero, se intentan usar monedas de 5 unidades si la parte entera del importe es múltiplo de 5; posteriormente, se intentan usar monedas de 2 unidades si es múltiplo de 2; y finalmente, si no se da nada de lo anterior, se usan `(importe div 100)` monedas de 1 unidad. De esta manera, siempre vamos a completar la parte entera del importe usando tantas veces como sea necesario un único tipo de moneda de unidades. Al ejecutar esta idea para la entrada de ejemplo (122233367 céntimos), obtengo **680 soluciones en 2s 88ms**.

1.5. Apartado 1 con codificación flotante

Usando la codificación flotante, MiniZinc nos da el siguiente error: "*Error: Gecode: Float::linear: Number out of limits*" (1). Esta excepción la lanza el solver Gecode cuando intenta manejar una **operación de números decimales que excede los límites de precisión admitidos** por su motor de propagación de restricciones en coma flotante. Este problema deriva de la línea de código `constraint sum(i in POS)(cantidades[i] * monedas[i]) = importe`, donde el solver intenta encontrar una combinación exacta de monedas que iguale al importe.

Sin embargo, si usamos operaciones que multiplican variables flotantes, el espacio de búsqueda se amplía a cantidades no razonables y Gecode lo rechaza por ser demasiado grande y poco preciso.

De hecho, la restricción anterior puede fallar inconsistentemente por diferencias mínimas de representación, ya que los flotantes en binario no se pueden representar con exactitud y se pueden dar casos de que 0,12999 $\neq$ 0,13. Por todo ello, este problema del cambio de monedas debe resolverse con codificación entera.

```
Running Ejercicio1e.mzn
Error: Gecode: Float::linear: Number out of limits
=====ERROR=====
%%%mzn-stat: nSolutions=0
%%%mzn-stat-end
Process finished with non-zero exit code 1.
Finished in 218msec.
```

Figura 1: Error obtenido al usar la codificación flotante

2. Puzzle lógico

Al resolver este problema de satisfacción de restricciones obtenemos dos posibles soluciones en 334 ms. Podemos ver las soluciones encontradas en la figura 2a.

```
Running Ejercicio2.mzn
Madrid: Cristina y Marta 10 años
Barcelona: Jose y Pedro 25 años
Sevilla: Claudia y Marcos 5 años
Lyon: Lucia y Juan 30 años
Parma: Sofia y Pablo 20 años
----------
Madrid: Cristina y Marta 10 años
Barcelona: Lucia y Pedro 25 años
Sevilla: Claudia y Marcos 5 años
Lyon: Juan y Jose 30 años
Parma: Sofia y Pablo 20 años
----------
==========
```

(a) Soluciones encontradas para el ejercicio 2

```
Running Ejercicio3.mzn
Matriz binaria completada:

1 1 0 0 1 0 1 0 1 1 0 0
0 1 1 0 0 1 0 1 0 0 1 1
1 0 0 1 1 0 1 0 1 0 1 0
1 1 0 0 1 1 0 0 1 1 0 0
0 0 1 1 0 0 1 1 0 0 1 1
0 1 1 0 0 1 1 0 0 1 1 0
1 0 0 1 1 0 0 1 1 0 0 1
1 0 1 1 0 0 1 0 0 1 0 1
0 1 0 0 1 1 0 1 0 1 1 0
0 0 1 1 0 1 1 0 1 0 1 0
1 0 1 0 1 0 0 1 0 1 0 1
0 1 0 1 0 1 0 1 1 0 0 1
----------
==========
```

(b) Solución encontrada para el ejercicio 3

Figura 2: Salidas de MiniZinc para los ejercicios 2 y 3

3. Problema de la tabla binaria

Según el código realizado, este problema tiene una única solución, encontrada en 462 ms. Es la que muestra la figura 2b.

4. Problema de los cuadrados

X	S	$\|S\|$	Runtime
4	UNSATISFIABLE	UNSATISFIABLE	UNSATISFIABLE
5	{4, 3}	2	328ms
6	UNSATISFIABLE	UNSATISFIABLE	UNSATISFIABLE
7	{6, 3, 2}	3	339ms
8	UNSATISFIABLE	UNSATISFIABLE	UNSATISFIABLE
9	{6, 5, 4, 2}	4	317ms
10	{7, 5, 4, 3, 1}	5	327ms
15	{11, 7, 5, 4, 3, 2, 1}	7	324ms
30	{15, 13, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}	13	324ms
40	{23, 15, 14, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}	15	377ms
50	{22, 20, 17, 16, 15, 14, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}	18	543ms
55	{26, 23, 18, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}	19	818ms
60	{23, 22, 21, 19, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}	21	1788ms
70	{24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}	24	2543ms
80	{30, 29, 27, 24, 22, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}	25	11074ms
90	{36, 32, 28, 25, 24, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}	27	34893ms
100	{38, 36, 30, 28, 26, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}	29	1m 10s
110	{36, 35, 32, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}	32	3m 15s
120	{38, 34, 33, 32, 31, 30, 29, 28, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1}	34	5m 16s

Cuadro 4: Resultados del Problema de los cuadrados

He añadido varios valores de X para poder realizar mejor el estudio de la complejidad del problema. Viendo la gráfica 3 podemos decir que este problema tiene **complejidad exponencial**, pues dicha función sigue la tendencia de una exponencial. Por tanto, no es un problema escalable. Para los valores de X de 1 o 2 cifras el tiempo empleado es menor de 1 minuto; mientras que para los valores de 3 cifras ya se supera la barrera del minuto y comienza a ascender vertiginosamente.

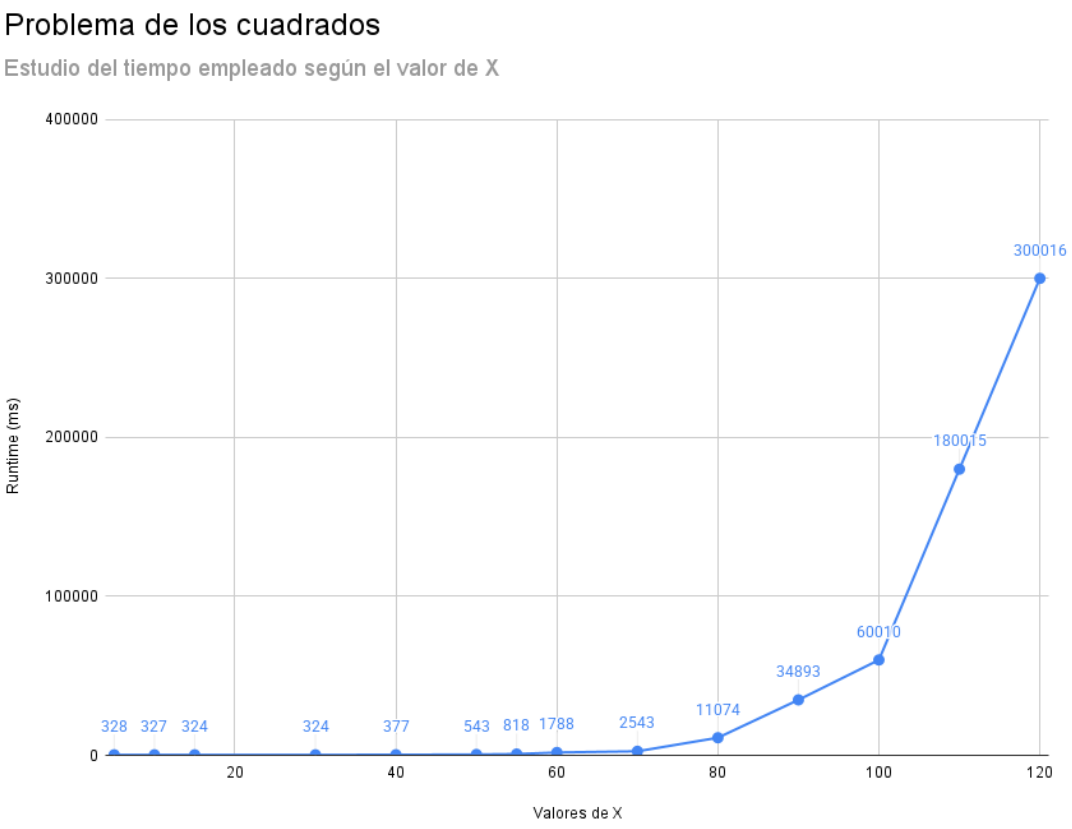

Figura 3: Estudio de la complejidad computacional del problema de los cuadrados

5. Problema de planificación de tareas

5.1. Problema optimización de restricciones (COP)

Con la implementación mandada, he obtenido la solución de la figura 5. Como la última tarea se ha terminado en el día 26, concluimos que **la duración mínima de la construcción del DeLorean es de 26 días**. Esta planificación encontrada se detalla en el diagrama de Gantt de la figura 4.

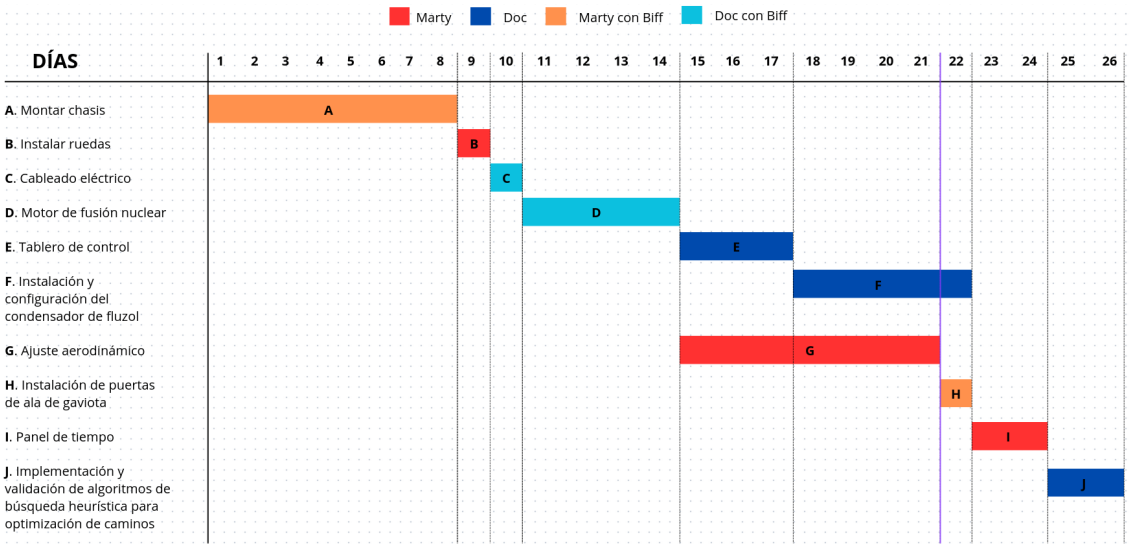

Figura 4: Diagrama de Gantt con la planificación encontrada

```
Running Ejercicio5a.mzn
  "Montar chasis              " inicio día 1 fin día 8. Asignada a Marty con apoyo de Biff
  "Instalar ruedas            " inicio día 9 fin día 9. Asignada a Marty
  "Cableado eléctrico         " inicio día 10 fin día 10. Asignada a Doc con apoyo de Biff
  "Motor de fusión nuclear    " inicio día 11 fin día 14. Asignada a Doc con apoyo de Biff
  "Tablero de control         " inicio día 15 fin día 17. Asignada a Doc
  "Inst. condensador de fluzo " inicio día 18 fin día 22. Asignada a Doc
  "Ajuste aerodinámico        " inicio día 15 fin día 21. Asignada a Marty
  "Inst. puertas de gaviota   " inicio día 22 fin día 22. Asignada a Marty con apoyo de Biff
  "Panel de tiempo            " inicio día 23 fin día 24. Asignada a Marty
  "Impl. alg. de búsqueda heu." inicio día 25 fin día 26. Asignada a Doc
  ----------
  ==========
```

Figura 5: Solución obtenida en el ejercicio 5a

5.2. Problema satisfacción de restricciones (CSP)

En esta versión del problema, la anterior variable a minimizar `duracionMontaje` la quitamos y la ponemos como constante, asignándole el valor mínimo encontrado (`int: duracionMontaje = 26`). Y ahora resolvemos en el modo satisfacción de restricciones para ver cuántas soluciones de asignación de tareas hay que acaben el DeLorean en 26 días. En concreto, obtenemos **2 posibles soluciones**, las indicadas en la figura 6.

```
Running Ejercicio5b.mzn
  "Montar chasis              " inicio día 1 fin día 8. Asignada a Marty con apoyo de Biff
  "Instalar ruedas            " inicio día 9 fin día 9. Asignada a Marty
  "Cableado eléctrico         " inicio día 10 fin día 10. Asignada a Doc con apoyo de Biff
  "Motor de fusión nuclear    " inicio día 11 fin día 14. Asignada a Doc con apoyo de Biff
  "Tablero de control         " inicio día 15 fin día 17. Asignada a Doc
  "Inst. condensador de fluzo " inicio día 18 fin día 22. Asignada a Doc
  "Ajuste aerodinámico        " inicio día 15 fin día 21. Asignada a Marty
  "Inst. puertas de gaviota   " inicio día 22 fin día 22. Asignada a Marty con apoyo de Biff
  "Panel de tiempo            " inicio día 23 fin día 24. Asignada a Marty
  "Impl. alg. de búsqueda heu." inicio día 25 fin día 26. Asignada a Doc
  ----------
  "Montar chasis              " inicio día 1 fin día 8. Asignada a Marty con apoyo de Biff
  "Instalar ruedas            " inicio día 9 fin día 9. Asignada a Marty
  "Cableado eléctrico         " inicio día 10 fin día 10. Asignada a Doc con apoyo de Biff
  "Motor de fusión nuclear    " inicio día 11 fin día 14. Asignada a Doc con apoyo de Biff
  "Tablero de control         " inicio día 15 fin día 17. Asignada a Doc
  "Inst. condensador de fluzo " inicio día 18 fin día 22. Asignada a Doc
  "Ajuste aerodinámico        " inicio día 15 fin día 21. Asignada a Marty
  "Inst. puertas de gaviota   " inicio día 22 fin día 22. Asignada a Marty con apoyo de Biff
  "Panel de tiempo            " inicio día 23 fin día 24. Asignada a Doc
  "Impl. alg. de búsqueda heu." inicio día 25 fin día 26. Asignada a Doc
  ----------
  ==========
```

Figura 6: Soluciones obtenidas en el ejercicio 5b

6. Problema de conformación de tribunales de TFG

Con la implementación proporcionada he obtenido **320 soluciones**, de las cuales debe haber muchas duplicadas. Tras hacer una revisión de las mismas, he llegado a la conclusión de que no hay simetrías por estar escritos los profesores en otro orden, pues siempre siguen un orden ascendente, al igual que con la lista de los TFGs que evalúa cada tribunal.

Sí que he detectado que se repiten soluciones intercambiando el tribunal 1 por el 2, manteniendo los mismos profesores, departamentos, días y TFGs a evaluar. En la figura 7 se muestran ejemplos de estas soluciones simétricas. Me ha quedado pendiente corregir dichas simetrías y comprobar que realmente no haya más, porque sospecho que obtengo demasiadas soluciones únicas válidas, aún quedándose en 160 (quitando la simetría mencionada).

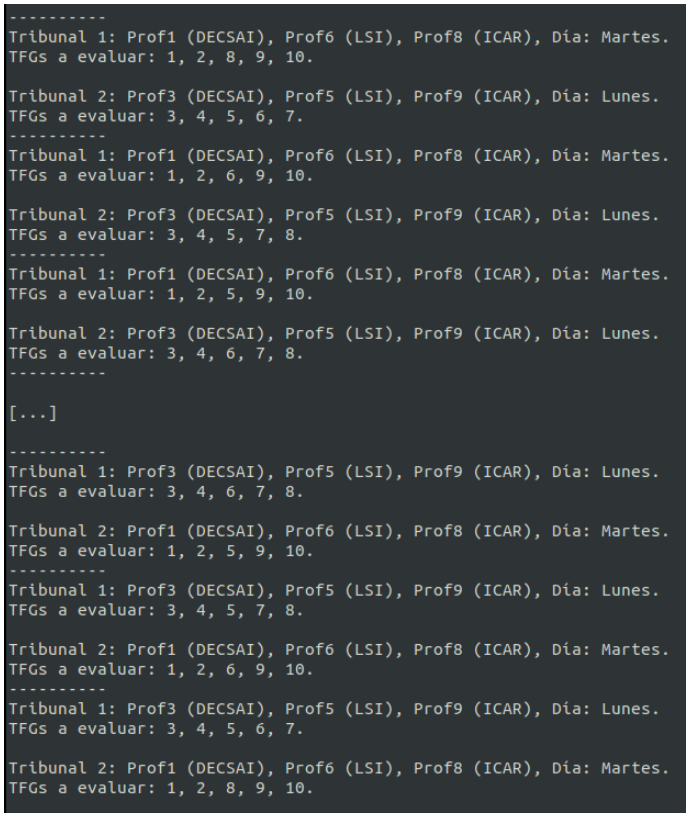

Figura 7: Ejemplos de soluciones simétricas encontradas en el ejercicio de los tribunales