Universidad del Valle

Facultad de Ingeniería

Escuela de Ingeniería de Sistemas y Computación

Programación con Restricciones

Informe taller: Planificación de los ensayos en una telenovela

Integrantes:

Hector Diaz Hurtado - 2310001
Lenin Carabali Moreno - 2310025
Juan José Moreno Jaramillo - 2310038
Brayan Gomez Muñoz - 2310016

Fecha:

17 de Julio 2025

Índice

1. Introducción

El presente informe, elaborado en el marco del curso de Programación por Restricciones, se centra en la planificación de los ensayos de la telenovela *Desenfreno de Pasiones*, un proceso en el que cada intérprete debe permanecer en el set desde su primera hasta su última escena y en el que los tiempos de inactividad suponen un costo significativo para la producción. Para abordar este desafío se propuso un modelo 1 básico, en el que se formalizaron los datos de entrada —actores, escenas, tarifas y duraciones— y se definió una función objetivo orientada a minimizar el coste total de los periodos de espera de los actores.

A partir de esta base, se diseñó un modelo extendido que incorpora restricciones adicionales de disponibilidad y compatibilidad entre intérpretes: algunos actores estrella limitan su estancia máxima en el estudio y se busca reducir al mínimo las coincidencias simultáneas de aquellos con dinámicas conflictivas, sin apartar el foco de la reducción de costos. Ambos modelos fueron implementados en MiniZinc, se sometieron a distintas estrategias de búsqueda y se probaron con diversos solvers, y finalmente sus resultados se compararon para evaluar el desempeño.

2. Parte 1- Modelo básico

2.1. Descripción del modelo

Se modela la planificación de los ensayos de la telenovela *Desenfreno de Pasiones* como un problema de permutación de escenas, de modo que los periodos de inactividad de los actores queden minimizados.

2.1.1. Parámetros de entrada

- **ACTORES:** Conjunto de intérpretes que participan en los ensayos.
  - **Dominio:** $ACTORES = \{\text{Actor}_1, \ldots, \text{Actor}_m\}$, con $m \geq 1$.
  - **Uso en el modelo:** Indexa las filas de la matriz `Escenas` y el vector de tarifas.
- **Escenas:** Matriz de dimensión $|ACTORES| \times (n\_escenas + 1)$ que agrupa información de participación y coste de cada actor.
  - **Dominio:**

$$Escenas[a, s] \in \{0, 1\} \quad \forall\, a \in ACTORES,\ s = 1, \ldots, n\_escenas;$$

$$Escenas[a, n\_escenas + 1] \in N \quad \forall\, a \in ACTORES.$$

  - **Uso en el modelo:**
    - Columnas $1 \ldots n\_escenas$: indican la presencia binaria del actor en cada escena.
    - Columna $n\_escenas+1$: tarifa por unidad de tiempo, utilizada en el cálculo del coste.
- **Duracion:** Vector de longitud $n\_escenas$ con la duración de cada escena en unidades de tiempo.
  - **Dominio:**

$$Duracion[s] \in N,\ Duracion[s] \geq 1, \quad s = 1, \ldots, n\_escenas.$$

  - **Uso en el modelo:** Cada valor se multiplica por la tarifa del actor para computar el coste de inactividad entre su primera y última aparición.

2.1.2. Constantes Auxiliares

- **NumActores:** Cantidad de intérpretes.

$$NumActores = |ACTORES|, \quad NumActores \in N,\ NumActores \geq 1.$$

- **n_escenas:** Número de escenas a ordenar.

$$n_{\text{escenas}} = |Duracion|, \quad n_{\text{escenas}} \in N,\ n_{\text{escenas}} \geq 1.$$

- **Participa**$[a, s]$**:** Indica si el actor a interviene en la escena s.

$$Participa[a,s] = \begin{cases} true, & \text{si } Escenas[a,s] = 1, \\ false, & \text{si } Escenas[a,s] = 0, \end{cases} \quad \forall\, a \in ACTORES,\ s = 1, \ldots, n_{\text{escenas}}.$$

$$Participa[a,s] \in \{true, false\}.$$

- **Coste**$[a]$**:** Tarifa por unidad de tiempo del actor a.

$$Coste[a] = Escenas[a,\, n_{\text{escenas}} + 1], \quad Coste[a] \in N,\ \forall\, a \in ACTORES.$$

2.2. Variables

- $order[p]$: escena asignada a la posición p en la secuencia de ensayos.

$$order[p] \in \{1, \ldots, n\_escenas\}, \quad \forall p = 1, \ldots, n\_escenas.$$

- $invOrd[s]$: posición en la que aparece la escena s en la permutación.

$$invOrd[s] \in \{1, \ldots, n\_escenas\}, \quad \forall s = 1, \ldots, n\_escenas.$$

- $firstPos[a]$: primera posición de ensayo del actor a.

$$firstPos[a] \in \{1, \ldots, n\_escenas\}, \quad \forall a \in ACTORES.$$

- $lastPos[a]$: última posición de ensayo del actor a.

$$lastPos[a] \in \{1, \ldots, n\_escenas\}, \quad \forall a \in ACTORES.$$

- $costo\_actor[a]$: coste total abonado al actor a por el tiempo que permanece en el set, desde su primera hasta su última aparición:

$$costo\_actor[a] = \sum_{p=firstPos[a]}^{lastPos[a]} Duracion[order[p]] \times Escenas[a, n\_escenas{+}1], \quad \forall a \in ACTORES.$$

- $costo\_total$: suma de los costes de todos los actores:

$$costo\_total = \sum_{a \in ACTORES} costo\_actor[a].$$

2.3. Restricciones

1. **Sobreyectividad de la permutación.** Esta restricción asegura que cada escena sea asignada a alguna posición, de modo que no quede ninguna escena sin ubicar:

$$\{\ \text{order}(p) \mid p = 1, \ldots, n\ \} = \{1, \ldots, n\}.$$

2. **Inyectividad de la permutación.** Esta restricción garantiza que no existan dos posiciones distintas con la misma escena, asegurando la unicidad de la asignación:

$$\text{order}(p) \neq \text{order}(p') \quad \forall\, p, p' \in \{1, \ldots, n\},\ p \neq p'.$$

3. **Determinación de primera y última aparición.** Para cada actor a, se define la primera posición como el mínimo de las posiciones donde participa, y la última como el máximo:

$$firstPos(a) = \text{mín}\{\, invOrd(s) \mid P_{a,s} = True\}$$

$$lastPos(a) = \text{máx}\{\, invOrd(s) \mid P_{a,s} = True\}.$$

4. **Cálculo del coste individual.** El coste de inactividad de cada actor a se modela como la suma de las duraciones de las escenas entre su primera y última aparición, ponderadas por su tarifa:

$$costo\_actor(a) \;=\; \sum_{p=firstPos(a)}^{lastPos(a)} Duracion[\, order[p]\,] \;\times\; Escenas[a,\, n\_escenas + 1],$$

$$\forall\, a \in ACTORES.$$

5. **Coste total pagado a los actores.** Se define la variable

$$costo\_total \;=\; \sum_{a \in ACTORES} costActor(a).$$

que representa el pago total que la producción realiza a los intérpretes, incluyendo tanto el tiempo de actuación como el de espera.

2.4. Función objetivo

La **función objetivo** de este modelo es minimizar el **costo total pagado a los actores** mediante la determinación del orden óptimo de las escenas.

$$\text{minimizar } costo\_total$$

2.5. Pruebas

Para este proyecto se crearon varias instancias con el objetivo de evaluar el modelo, incluyendo también las instancias proporcionadas por el profesor como parte de la batería de pruebas, con el fin de verificar la correctitud de las soluciones generadas. Se diseñaron instancias pequeñas y otras de mayor tamaño, aumentando progresivamente el número de escenas o actores. Además, se utilizaron distintos solvers y estrategias de búsqueda para

cada archivo, con el propósito de analizar las diferencias entre las soluciones obtenidas en cada caso.

Para automatizar el proceso de evaluación, se desarrolló un archivo en Python que ejecuta las pruebas de manera sistemática. Los resultados fueron organizados y registrados en una hoja de cálculo, lo que permitió facilitar su análisis y comparación.

Para la evaluación del modelo, se emplearon diferentes combinaciones de estrategias de búsqueda y solvers. Entre las estrategias utilizadas se encuentran `(input_order, indomain_min)`, `(first_fail, indomain_median)`, `(first_fail, indomain_split)` y `(dom_w_deg, indomain_min)`. Estas estrategias permitieron observar el comportamiento del modelo bajo distintas formas de exploración del espacio de soluciones. Adicionalmente, se utilizaron los solvers `HiGHS`, `COIN-BC` y `Gecode` para resolver las instancias, lo que permitió comparar el rendimiento y la calidad de las soluciones generadas por cada uno.

2.5.1. Resultados batería de pruebas

Probamos HiGHS, COIN-BC y Gecode con las cuatro heurísticas `(input order, indomain min)`, `(first fail, indomain median)`, `(first fail, indomain split)` y `(dom w deg, indomain min)` en todas las instancias, de más pequeñas a más grandes.

- En los casos pequeños todos llegan al óptimo en milisegundos. Gecode recorre más nodos, COIN-BC se queda en un término medio e HiGHS explora muy poco para dar con la primera solución.
- A partir de tamaños medios COIN-BC sigue rápido pero ya empieza a mostrar timeouts, y Gecode se encarga de explorar más a fondo (a costa de tiempo). HiGHS sigue dando respuestas rápidas, aunque no siempre exhaustivas.
- En la instancia más dura (`Desenfreno2.dzn`, límite 600 s) ni Gecode ni COIN-BC lograron terminar.

Todos los datos de tiempo, nodos explorados y soluciones factibles están disponibles en la Tabla de Pruebas - Modelo 1.

2.5.2. Análisis de los resultados

Durante la evaluación observamos que, pese a emplear distintos solvers y estrategias de búsqueda, **todos convergieron en la misma solución óptima**, lo cual coincide con los resultados de referencia proporcionados por el profesor. Sin embargo, aunque el valor de la solución es idéntico en todas las ejecuciones, **el orden de asignación de escenas puede variar**. Esto demuestra que existen varias permutaciones equivalentes que satisfacen todas las restricciones sin alterar el óptimo, y que la selección de heurística (por ejemplo, en Gecode) influye en la secuencia en la que se descubren dichas soluciones.

Adicionalmente, se observaron diferencias significativas en el tamaño de los árboles de búsqueda y en el número de soluciones factibles reportadas por cada solver:

- `Gecode` recorre, en promedio, la mayor cantidad de nodos antes de alcanzar la solución óptima, generando también múltiples soluciones factibles en el proceso. Esto indica un análisis profundo del espacio de búsqueda y una tendencia a explorar exhaustivamente varias permutaciones válidas.

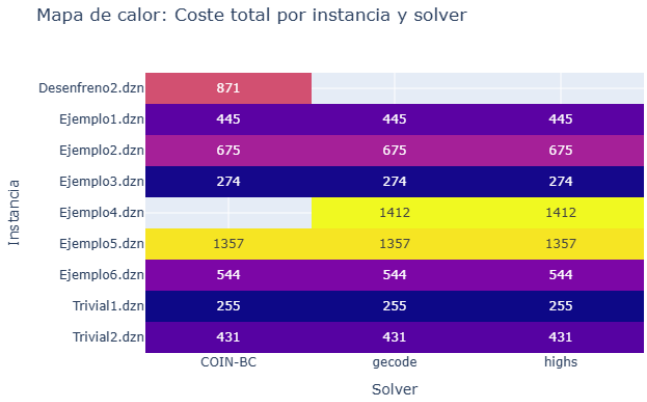

Figura 1: Valores de Costo total por Solver

- `COIN-BC` queda en un rango intermedio de nodos explorados, presentando un balance entre exhaustividad y rapidez, pero en instancias complejas puede llegar a timeout sin completar la búsqueda.
- `HiGHS`, por su parte, explora consistentemente un número reducido de nodos (decenas), lo que refleja un enfoque más conservador y rápido en la generación de las primeras soluciones factibles.

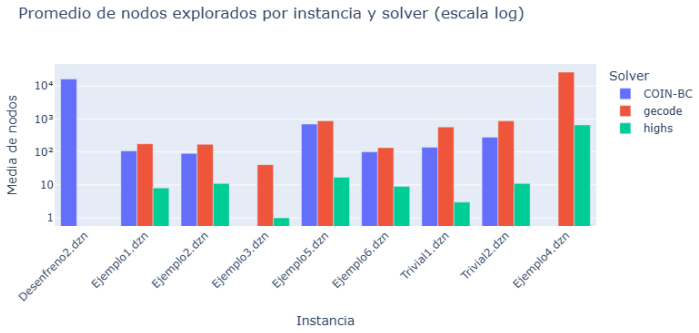

Figura 2: Nodos explorados

En términos de velocidad, `Gecode` sigue siendo muy competitivo en instancias pequeñas (resoluciones en milisegundos), aunque su tope de exploración crece sustancialmente en problemas de mayor escala. Para la instancia más compleja (`Desenfreno2.dzn`), con un timeout fijado en 600 s ni en un lapso de 1 h, ni `Gecode` ni `COIN-BC` lograron concluir la búsqueda dentro del límite establecido.

3. Parte 2 - Modelo extendido

3.1. Descripción general

Esta parte extiende el modelo básico presentado anteriormente, incorporando nuevas restricciones que permiten representar condiciones adicionales del problema de planificación de ensayos para la telenovela *Desenfreno de Pasiones*. El objetivo sigue siendo minimizar el costo total de los actores, pero ahora también se penaliza el tiempo compartido en el set entre ciertos pares de actores que deben evitar coincidir.

3.1.1. Parámetros de entrada

- **Disponibilidad:** Matriz $n \times 2$ con pares `[actor, tiempo_máximo]` que indica el tiempo máximo que el actor puede permanecer en el set.

$$Disponibilidad[i, 1] \in ACTORES, \quad Disponibilidad[i, 2] \in N_0$$

- **Evitar:** Matriz $m \times 2$ que indica pares de actores que deben evitar coincidir en el set.

$$Evitar[k, 1], Evitar[k, 2] \in ACTORES, \quad \forall\, k = 1, \ldots, m$$

3.1.2. Variables

- `Escenas_[`a`, `s`]`: Variable binaria que indica si el actor a participa en la escena ubicada en la posición s del orden actual.

$$Escenas\_[a, s] \in \{0, 1\}$$

- `tiempo_presencia_actor[`a`]`: Tiempo total que el actor a permanece en el set, calculado como la suma de las duraciones desde su primera hasta su última aparición.

$$tiempo\_presencia\_actor[a] \in N$$

- `tiempo_compartido[`k`]`: Tiempo en que los actores del par k están simultáneamente presentes en el set.

$$tiempo\_compartido[k] \in N$$

- `evitar_incumplida[`k`]`: Variable binaria que indica si el par de actores k incumple la restricción de no coincidir.

$$evitar\_incumplida[k] \in \{0, 1\}$$

- `costes[`a`]`: Vector de costos individuales de cada actor a, extraído de los datos de entrada.

$$costes[a] \in N$$

- `max_coste`: Valor máximo del vector de costos de los actores.

$$max\_coste \in N$$

- `actor_caro`: Índice del actor con el mayor costo de contratación.

$$actor\_caro \in ACTORES$$

- `promedio_posiciones`: Promedio de las posiciones (en el orden de escenas) en las que aparece el actor más caro.

$$promedio\_posiciones \in N$$

- `objetivo`: Función objetivo extendida, que combina el costo total con una penalización por tiempo compartido no deseado.

3.1.3. Restricciones

1. **Cálculo del tiempo de presencia de los actores:** Se calcula como la suma de la duración de todas las escenas entre la primera y última aparición del actor, participen o no en cada escena.

$$tiempo\_presencia\_actor[a] = \sum_{p=\texttt{firstPos}[a]}^{\texttt{lastPos}[a]} Duracion[order[p]]$$

2. **Restricción de disponibilidad máxima:** Si el actor tiene un tiempo máximo especificado, no debe superarlo.

$$\text{si } Disponibilidad[i,2] > 0, \quad \texttt{tiempo\_presencia\_actor}[i] \leq Disponibilidad[i,2]$$

3. **Cálculo de participación real:** Define si un actor aparece realmente en la escena ubicada en la posición j del orden.

$$Escenas\_[a,j] = Participa[a, order[j]]$$

4. **Tiempo compartido entre actores que deben evitarse:** Se acumula el tiempo en el que ambos actores del par k están presentes simultáneamente en el set.

$$tiempo\_compartido[k] = \sum_{j} 1_{j \in [FP_a, LP_a] \cap [FP_b, LP_b]} \cdot Duracion[order[j]]$$

5. **Activación del incumplimiento de coincidencia:** Se activa cuando las ventanas de presencia de dos actores del par k se solapan.

$$evitar\_incumplida[k] = 1 \iff \texttt{firstPos}[a] \leq \texttt{lastPos}[b] \land \texttt{firstPos}[b] \leq \texttt{lastPos}[a]$$

3.1.4. Restricción de simetría

$$promedio\_posiciones = \frac{\sum_{s=1}^{n\_escenas} Participa[actor\_caro, s] \cdot invOrd[s]}{\sum_{s=1}^{n\_escenas} Participa[actor\_caro, s]}$$

Para evitar que el *solver* genere soluciones redundantes (específicamente aquellas que corresponden a la imagen "espejo" de otras ya encontradas), se incorporó una restricción adicional al modelo. Esta restricción obliga a que las escenas en las que participa el actor de mayor coste aparezcan, en promedio, en la mitad inicial de la secuencia. De esta forma, se descarta automáticamente la versión invertida de cada solución y se selecciona una única representación canónica dentro de cada familia de permutaciones equivalentes, sin alterar el valor óptimo del coste.

Los ejemplos fueron corridos con Highs, ya que fue el único solucionador que dio respuesta en todas las instancias, y por ello se decidió realizar las pruebas con este.

Los efectos observados al aplicar esta restricción fueron los siguientes:

- El valor del coste total se mantuvo igual en todas las instancias evaluadas, garantizando que no se descartaron soluciones válidas.
- En muchos casos, con la restricción de rompimiento de simetrías, el *solver* encontró exactamente la misma solución esperada que se tenía como referencia. En contraste, con el modelo original (sin la restricción). No siempre se generaba ese orden preferido, aunque sí se obtenían permutaciones equivalentes en términos de coste. Ejemplo:

```
----------
Orden de escenas: [8, 6, 4, 2, 7, 5, 1, 3]
Coste total: 1412
Detalles por actor:
  A1: Escenas [6..8] Coste = 80, Tiempo en estudio = 8
  A2: Escenas [2..4] Coste = 77, Tiempo en estudio = 7
  A3: Escenas [4..7] Coste = 117, Tiempo en estudio = 9
  A4: Escenas [2..8] Coste = 204, Tiempo en estudio = 17
  A5: Escenas [5..8] Coste = 150, Tiempo en estudio = 10
  A6: Escenas [2..5] Coste = 144, Tiempo en estudio = 9
  A7: Escenas [1..8] Coste = 360, Tiempo en estudio = 20
  A8: Escenas [1..6] Coste = 280, Tiempo en estudio = 14
----------
==========
```

Figura 3: Ejemplo 4 - Con Simetria

```
----------
Orden de escenas: [3, 1, 5, 7, 2, 4, 6, 8]
Coste total: 1412
Detalles por actor:
  A1: Escenas [1..3] Coste = 80, Tiempo en estudio = 8
  A2: Escenas [5..7] Coste = 77, Tiempo en estudio = 7
  A3: Escenas [2..5] Coste = 117, Tiempo en estudio = 9
  A4: Escenas [1..7] Coste = 204, Tiempo en estudio = 17
  A5: Escenas [1..4] Coste = 150, Tiempo en estudio = 10
  A6: Escenas [4..7] Coste = 144, Tiempo en estudio = 9
  A7: Escenas [1..8] Coste = 360, Tiempo en estudio = 20
  A8: Escenas [3..8] Coste = 280, Tiempo en estudio = 14
----------
==========
```

Figura 4: Ejemplo 4 - Sin Simetria

Se evidencia que, para algunos casos de estudio, la ejecución retorna órdenes simétricos con el mismo valor de costo. Por lo tanto, la restricción de simetría añadida,

ahora solo retorna el orden que prioriza las escenas del actor más costoso, evitando búsquedas innecesarias que conducen a soluciones espejo sin aportar mejoras.

- Se observó una notable reducción en el número de nodos explorados, lo que acortó el árbol de búsqueda y aceleró la resolución del problema.
- Las soluciones en forma de espejo dejaron de ser alternativas válidas, ya que ahora se elige la solución que respeta el orden definido por la restricción. Esto permite reducir la cantidad de órdenes distintos de escenas y consolidando una única solución representativa por cada conjunto simétrico.
- En instancias críticas que previamente alcanzaban o rozaban el tiempo límite (600 s), la búsqueda se completó en un tiempo considerablemente menor.

Métrica	**Casos mejoró**	**Total ejecuciones**	**% de mejora**
Menor número de nodos	20	40	50 %
Misma respuesta	40	40	100 %
Mismo orden de escenas	15	40	37.5 %
Respuesta “espejo”	5	40	12.5 %
Menor tiempo de resolución	29	40	72.5 %

Cuadro 1: Comparación de resultados sin y con restricción de simetría

Todas las pruebas realizadas están plasmadas en Tabla de pruebas - Modelo 1 - Rompimiento de Simetría.

3.2. Función objetivo

El nuevo objetivo considera tanto el costo total de los actores como una penalización proporcional al tiempo de coincidencia entre pares de actores que deben evitarse:

$$\text{minimizar } objetivo = costo\_total + 1000 \times tiempo\_compartido\_total$$

Este factor multiplicativo se utiliza como penalización fuerte para reducir la coincidencia no deseada entre los actores especificados.

3.2.1. Resultados batería de pruebas - Pt2

Para la Parte 2 incorporamos las restricciones de "evitar coincidencias" y las anotaciones para romper simetrías, además de registrar el "tiempo compartido" de los actores conflictivos. Los puntos clave son:

- El valor del coste total se mantiene exactamente igual en todas las corridas, independientemente del solver o la heurística.
- Gecode muestra múltiples permutaciones de escenas válidas, mientras que HiGHS y COIN-BC suelen devolver siempre la misma asignación.
- En instancias grandes, Gecode y Coin-BC a veces agota el límite de 600 s y no termina, pero HiGHS sí lo hace con éxito de forma consistente.
- El número de nodos explorados difiere: Gecode genera árboles muy ramificados, con muchos candidatos intermedios; HiGHS y COIN-BC exploran de modo más compacto.

Todos los datos de tiempos, nodos explorados y conteo de permutaciones están en la Tabla de pruebas - Modelo 2.

3.2.2. Análisis de los resultados - Pt2

El valor del *coste total* y el *tiempo compartido* permanece invariable sin importar qué solver o heurística empleemos, ya que las nuevas restricciones impiden combinaciones libres. Sin embargo, **Gecode** explora más espacio de búsqueda (evidenciado en un mayor número de nodos explorados).y, además, notamos que las estrategias de búsqueda puede cambiar el orden de las escenas. Para ejemplificarlo, hemos utilizado la heurística `(input_order, indomain_min)`, en la cual se observan variaciones especialmente notables:

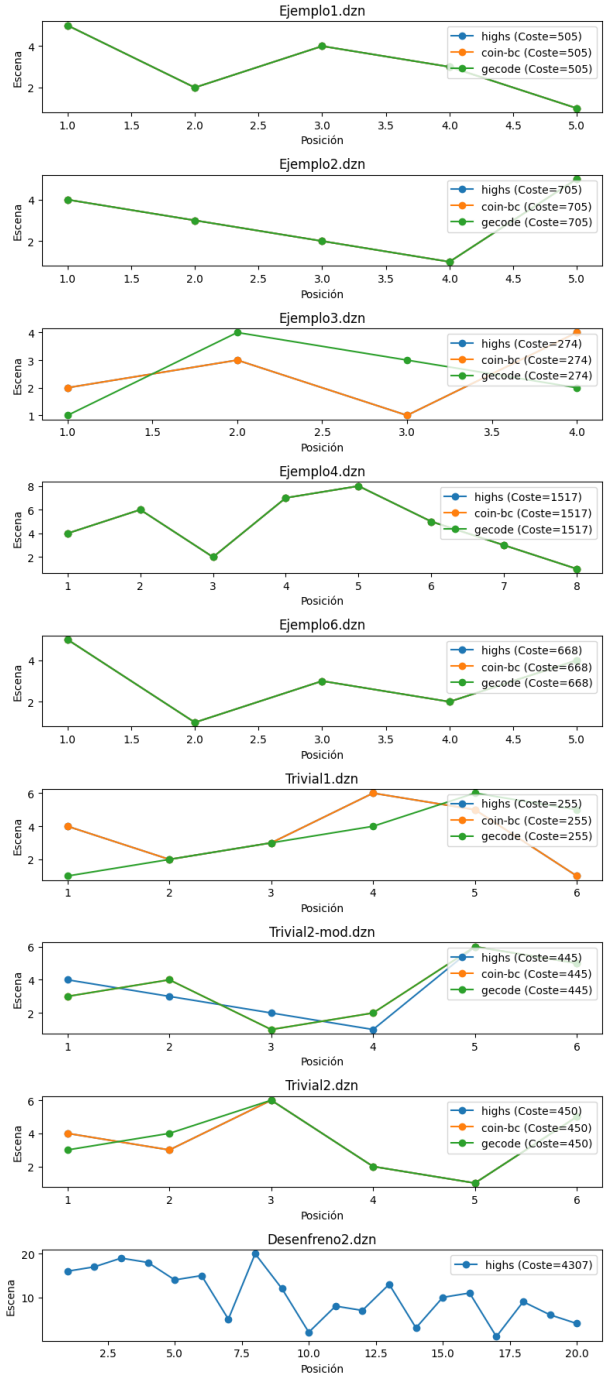

Figura 5: Variaciones de permutaciones con heurística `input_order, indomain_min`.

Al probar la instancia Desenfreno2.dzn, vimos que solo HiGHS termina en unos 540 s (explora 4 307 nodos), mientras que Gecode y COIN-BC hacen timeout (Mayor a 600 s). Notamos de nuevo que, aunque Gecode sea rápido en casos pequeños y medianos, su enfoque excesivamente exhaustivo lle es contraproducente en problemas muy grandes, donde el estilo más de HiGHS le permite llegar al final.

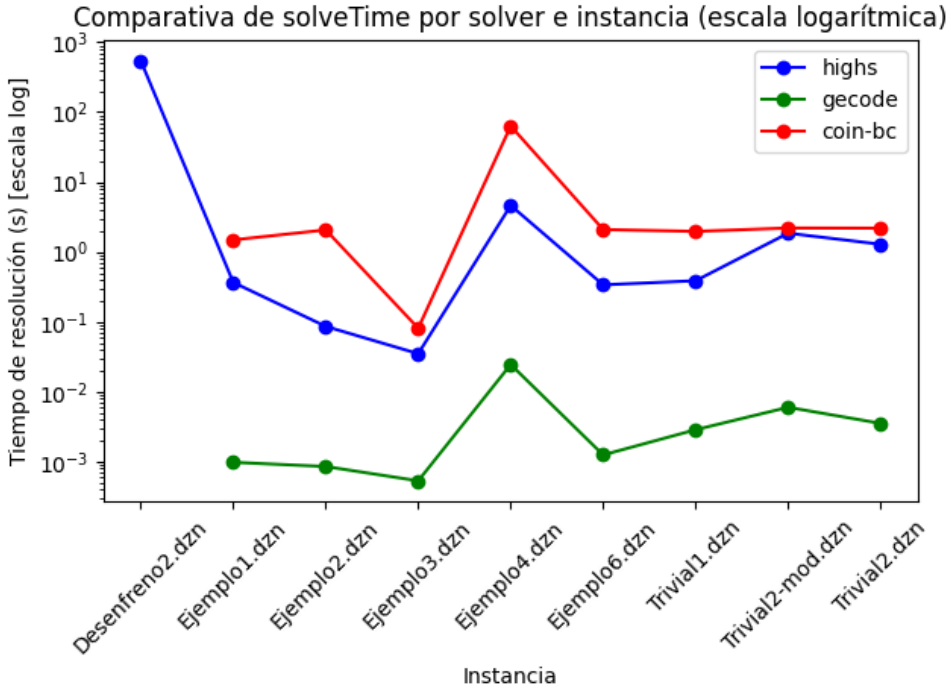

Figura 6: Tiempos

4. Conclusiones

- La aplicación de este paradigma ofrece una perspectiva interesante para abordar diferentes tipos de problemas demostrando que técnicas de optimización y entendimiento de dominios y limites se pueden adaptarse a infinidad de escenarios que normalmente no asociaríamos con modelos formales de programación. Se nota bastante en la construcción de un modelo ya que de manera relativamente abstracta se representa la búsqueda de valores posibles para responder y/o resolver diferentes problemas.

- Observamos que, aunque todas las ejecuciones alcanzan el mismo coste óptimo, la estrategia de búsqueda puede modificar el orden en que aparecen las escenas. Esto es importante porque, en un entorno real, diferentes permutaciones equivalentes podrían implicar diferencias logísticas (por ejemplo, orden de montaje o cambios de vestuario, etc).

- Cada solver muestra un comportamiento característico en cuanto a profundidad de exploración y rapidez. Es recomendable evaluar varios solucionadores para cada instancia y ajustar límites de tiempo según el perfil de cada uno.

- La ruptura de simetrías reduce el número de nodos explorados y en algunos casos acelera la resolución sin descartar soluciones óptimas. Respecto a este tipo de restricciones, pensamos que, no debe imponerse un orden fijo de respuesta para evitar rompimiento de simetrías, eso según la evaluación que hicimos en las pruebas al imponer dicha restricción.

- Aunque se evaluaron múltiples estrategias de búsqueda, no se identificó una claramente superior, dado que las diferencias como en nodos explorados y tiempos de ejecución resultaron pequeñas; las variaciones más significativas se atribuyen al cambio de solver. Si bien Gecode combinado con first fail e indomain median obtuvo mejor o peor desempeño en ciertos problemas concretos, para este casos explorados y con nuestros ejecuciones, evidenciamos que la selección del solver ejerce un impacto más determinante en las métricas de exploración que la elección de la heurística específica.