Planificación de los ensayos en una telenovela

1st Alexander González Cardenas
Escuela Ciencias de la computación
Universidad del Valle
Cali, Colombia
gonzalez.alexander@correounivalle.edu.co

2nd Carlos Esteban Murillo
Escuela Ciencias de la computación
Universidad del Valle
Cali, Colombia
murillo.carlos@correounivalle.edu.co

***Resumen*—En el presente trabajo se realizo el estudio de la grabación de una novela, con el cual se busca resolver en que orden se deben de organizar las escenas de tal forma que el productor de la novela obtenga un ahorro en el costo, puesto que cada actor cobra por unidad de tiempo que se encuentre en el set grabando una escena y su duración. Para este problema se hicieron varios modelos usando el estilo de programación por restricciones, se exploraron las simetrías que guarda el problema, se hicieron comparaciones entre solvers, y estrategias de búsqueda. Finalmente se uso el lenguaje de alto nivel `MiniZinc` para la implementación.**

***Palabras clave*—COP, MiniZinc, Orden de escenas**

I. Introducción

La programación por restricciones es una forma alternativa de resolución de problemas, en particular aquellos que se encuentran en los conocidos ***NP-Hard***, en los que se ofrece una solución que se considera "razonable" para cierto dominio de estudio de ese problema. Esta se sustenta en la combinación de técnica de razonamiento y computación. La idea central radica en pensar el mundo como una serie de variables cada una con un cierto dominio y las restricciones sobre estas variables [1].

II. Caracterización del problema

En este trabajo se trata de resolver el problema de los ensayos en una telenovela. Es bien sabido que las escenas de las telenovelas no necesariamente se ensayan en el orden en que se van a emitir (en algunas telenovelas el orden de las escenas se podría variar de cualquier manera sin que la trama sufriera, porque no la hay). No todos los actores participan en todas las escenas. Como a los actores les pagan por el tiempo total en que deben estar en el set, es muy bueno no dejarlos esperando después de una escena en que participan durante varias en que no lo hacen, hasta que llegue otra en que vuelven a participar. Para simplificar, supondremos que se conoce de antemano el número de escenas que componen la telenovela (lo que en Colombia es falso porque la duración de una telenovela puede ser infinita). El cuadro siguiente muestra un ejemplo de la participación en escenas para la telenovela "Desenfreno de Pasiones". Para efectos prácticos utilizaremos "Desenfreno de Pasiones" para explicar detalladamente el trabajo. Para cada actor, un

Escena	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Costo
Actor 1	1	1	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	10
Actor 2	1	1	1	0	0	0	1	1	0	1	0	0	1	1	1	0	1	0	0	1	4
Actor 3	0	1	1	0	1	0	1	1	0	0	0	0	1	1	1	0	0	0	0	0	5
Actor 4	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	5
Actor 5	0	1	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	1	1	1	5
Actor 6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	40
Actor 7	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	4
Actor 8	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	20
Durac.	2	1	1	1	1	3	1	1	1	2	1	1	2	1	2	1	1	2	1	1	

Figura 1: Escenas de desenfreno de pasiones

uno en la columna de la escena i indica que ese actor debe ensayar esa escena. Por ejemplo, el actor 6 (Narciso Perfiles, venezolano) participa en las escenas de la 14 a la 18. La última columna indica lo que cobra cada actor (en cientos de miles) por cada unidad de tiempo que le toque estar en el estudio. La última fila establece la duración de cada escena. Por ejemplo, la escena 6 dura 3 unidades de tiempo. El problema consiste en encontrar el orden en que se deben ensayar (o filmar) las escenas de modo que se minimice el costo total que se debe pagar a los actores. Hay que tener en cuenta que un actor debe estar presente al comienzo de la primera escena que le toca y solamente puede salir después de la ultima que le toca. Por ejemplo, si la secuencia de escenas se programara en el orden en que están en la tabla, el actor 3 debería estar en el estudio desde la escena 2 hasta la escena 15, o sea durante $1+1+1+1+3+1+1+1+2+1+1+2+1+2 = 19$, y cobraría $19 \times 5 = 95$, o sea \$9,500,000 COP.

III. Primer formalización del problema

Sea E el conjunto de todas las escenas que se desean grabar para la novela tal que $e \in E$ y sea A el conjunto

de actores que hay disponibles para grabar la novela tal que $a \in A$.

III-A. Parámetros

- Grabación$_{(e,a)}$ representa sí el actor a graba la escena e (se representa así: $1 \equiv$ Sí graba y $0 \equiv$ No graba)
- d_e representa la duración de la escena e.
- Costo$_a$ representa el costo por unidad de tiempo para el actor a.

III-B. Variables

- Sol$_i$ representa el orden de grabación i para cierta escena e, donde $i \in [1..\text{Card}(E)]$.

III-C. Dominio

- Sol$_i \in E$

III-D. Definición de Funciones

- UperBound$(K) \equiv$ Regresa el índice i en Sol$_i$ de la última escena del actor K
- LowerBound$(K) \equiv$ Regresa el índice i en Sol$_i$ de la primera escena del actor K

III-E. Restricciones

1. $\{\text{Sol}_i \neq \text{Sol}_j \,|\, \forall i, j \in [1..\text{Card}(E)] \wedge i \neq j\}$ (Es decir no se pueden grabar dos escenas en un mismo espacio de tiempo, es decir el mismo día).
2. $$\text{UperBound}(\beta) - \text{LowerBound}(\beta) + 1 \quad = \quad \sum_{e \in E} \text{Grabación}_{(e,\beta)}$$
   (Es decir la duración total debe ser tal que se grabe la mayor cantidad de escenas sucesivas posibles para ese actor, deben de quedar una seguida de la otra. Donde β corresponde al actor con el mayor costo posible en toda la grabación[1])
3. $$\text{UperBound}(\alpha) - \text{LowerBound}(\alpha) + 1 \quad \geq \quad \sum_{e \in E} \text{Grabación}_{(e,\alpha)}$$
   (Es decir la duración total debe de ser tal que se grabe la mayor cantidad de escenas posibles, con la diferencia que como se trata del menor se puede alargar mas, para ese actor, pueden quedar una seguida de la otra o alargarse. α corresponde al actor con el menor costo posible en toda la grabación[2])

III-F. Función Objetivo

Minimizar:

$$\sum_{a \in A} \sum_{k \in P(a)} d_k \times \text{Costo}_a$$

Donde $P(.)$ es una función tal que retorna el intervalo que representa el mapeo del correspondiente e de "." al momento desde donde inicia la grabación de su primera escena hasta la última que le toca en Sol$_i$.

IV. Implementación del primer modelo

IV-A. Ejemplo Trivial

IV-A1. Datos de entrada:

```
ACTORES = {Actor1, Actor2, Actor3}

Escenas = [| 0,1,0,0,0,0,10
           | 0,1,1,1,0,1,20
           | 0,0,0,0,1,1,15|]

Duracion = [2,1,1,1,3,4]
```

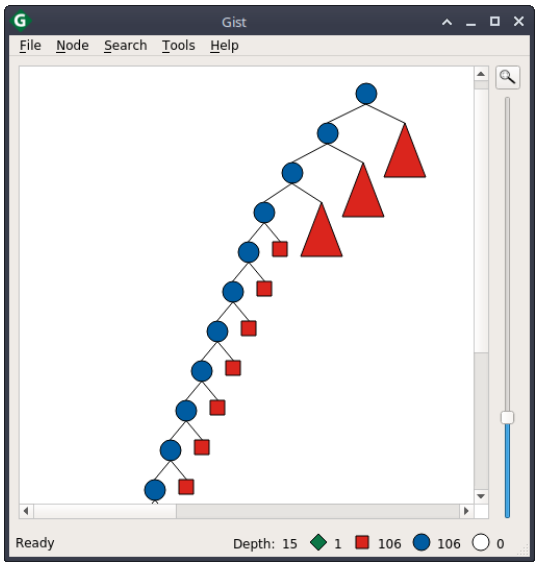


Figura 2: Árbol de búsqueda para el ejemplo Trivial

IV-A2. Análisis de resultados:

Para este ejemplo se puede ver que la búsqueda se hace profundidad (fig. 2), vale aclarar que las restricciones 2 y 3 se pueden ver

[1]En la implementación en `MiniZinc` se obtiene por medio de una función auxiliar llamada "getBigger"

[2]En la implementación en `MiniZinc` se obtiene por medio de una función auxiliar llamada "getSmaller"

como *redundantes* puesto que las soluciones parciales mostradas en el inspector de `MiniZinc` muestran una permutación de cierta instancia del problema, por lo que estas restricciones evitan una búsqueda en anchura con soluciones que posiblemente no se desean (fig. 3). Claro esta, que estas deducciones son meramente aplicables en el contexto de este problema y para reducir el espacio de búsqueda se imponen, como ya se ha dicho, las restricciones 2 y 3. Porque solo con 1 seria suficiente para resolver el problema.

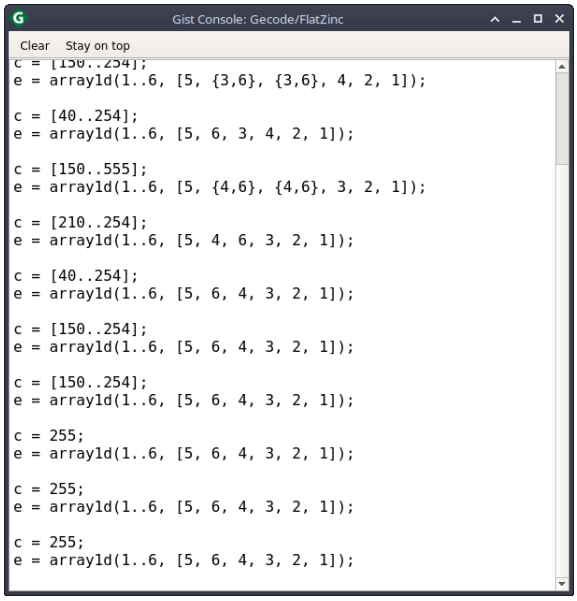


Figura 3: Inspector de solución para ejemplo trivial

IV-A3. Interpretación:

```
Sol = [5, 6, 4, 3, 2, 1]
Costo Total = 255
```

Esto quiere decir que la escena 5 se graba primero, luego se graba la 6 y así sucesivamente, también que el mejor costo total fue 255. Se puede observar que de la primera

Cuadro I: Tabla inicial de escenas para Trivial

Escenas	1	2	3	4	5	6	Costo
Actor 1	0	1	0	0	0	0	10
Actor 2	0	1	1	1	0	1	20
Actor 3	0	0	0	0	1	1	15

tabla (Cuadr. I) a la segunda tabla (Cuadr. II) se graban todas las escenas del actor 2 de forma sucesiva, como era de esperarse, así mismo el actor 1 se quedo hasta el final. Ahora bien otra solución valida es

`[1,4,3,2,6,5]`

Cuadro II: Tabla de escenas solución de Trivial

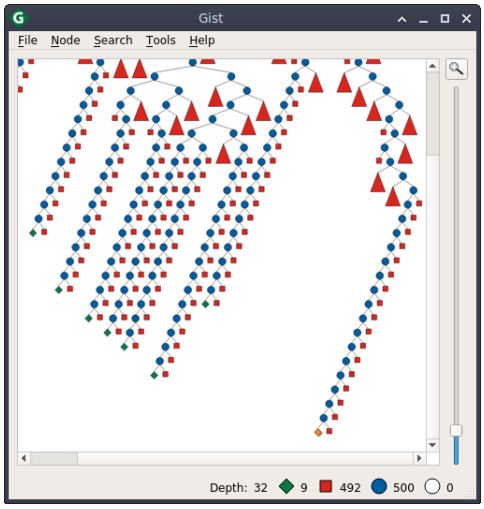
Escenas	5	6	4	3	2	1	Costo
Actor 1	0	0	0	0	1	0	10
Actor 2	0	1	1	1	1	0	20
Actor 3	1	1	0	0	0	0	15

que corresponde a un simple cambio de lugar que no tiene un impacto en el costo, puesto que mover la primera escena (que nadie graba por cierto) simplemente hace que la suma del final cambie de lugar al principio, luego nuevamente queda junta de forma sucesiva las escenas del actor 2 y el actor 1 al solo tener una escena no genera impacto. Es fácil ver con este problema la posible eliminación de simetrías (cosa que se abordara en una posterior sección) y que las restricciones redundantes cumplen con el objetivo de dar el menor costo. La estrategia de búsqueda usada para este ejemplo fue *por defecto* de `MiniZinc`.

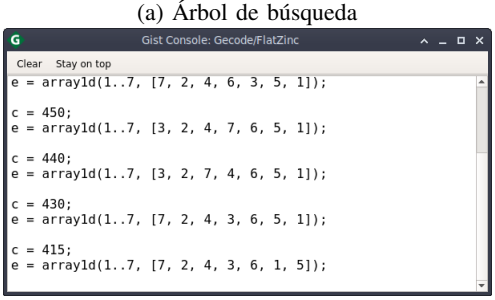
IV-B. Otros ejemplos

IV-B1. Datos de entrada: Ejemplo 2:

```
ACTORES = {Actor1, Actor2, Actor3,
           Actor4, Actor5};

Escenas = [| 0,1,0,0,0,0,1,10
           | 0,1,1,1,0,1,1,20
           | 1,1,1,1,0,1,0,5
           | 0,1,1,1,0,0,1,5
           | 1,0,0,0,1,1,0,15|];

Duracion = [2,1,1,1,3,4,2];
```

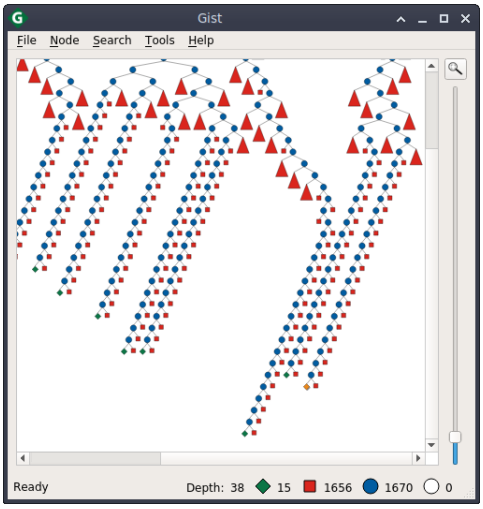


(a) Árbol de búsqueda

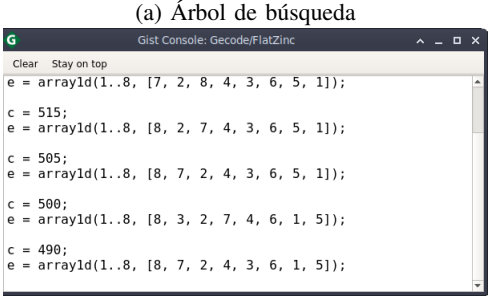


(b) Inspector de solución

Figura 4: Resultados para ejemplo 2



(a) Árbol de búsqueda



(b) Inspector de solución

Figura 5: Resultados para ejemplo 3

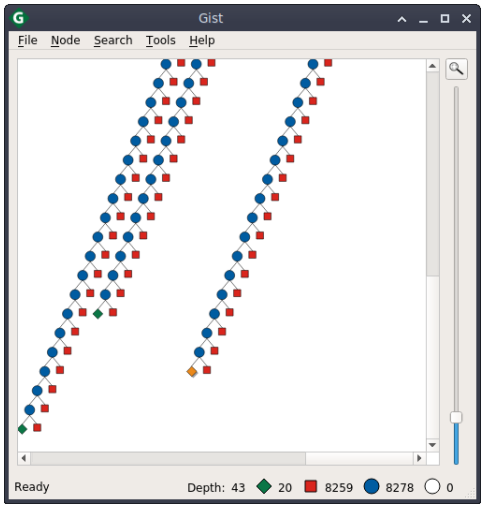
En este árbol de búsqueda se puede apreciar mas como impacta el tamaño de la entrada en el árbol de búsqueda respecto del Trivial. De manera similar se puede observar como por cada rama de un árbol se llega a un posible candidato para ser la solución óptima, sin embargo a comparación de Trivial se realizan mucho mas permutaciones.

IV-B2. Datos de entrada: Ejemplo 3:

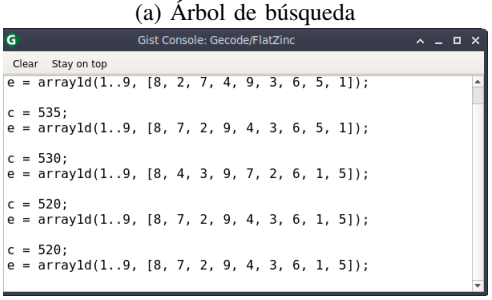
```
ACTORES = {Actor1, Actor2, Actor3,
           Actor4, Actor5};

Escenas = [| 0,1,0,0,0,0,1,0,10
           | 0,1,1,1,0,1,1,1,20
           | 1,1,1,1,0,1,0,0,5
           | 0,1,1,1,0,0,1,1,5
           | 1,0,0,0,1,1,0,0,15|];

Duracion = [2,1,1,1,3,4,2,3];
```

IV-B3. Datos de entrada: Ejemplo 4:

```
ACTORES = {Actor1, Actor2, Actor3,
           Actor4, Actor5};

Escenas = [| 0,1,0,0,0,0,1,0,0,10
           | 0,1,1,1,0,1,1,1,1,20
           | 1,1,1,1,0,1,0,0,1,5
           | 0,1,1,1,0,0,1,1,1,5
           | 1,0,0,0,1,1,0,0,0,15|];

Duracion = [2,1,1,1,3,4,2,3,1];
```

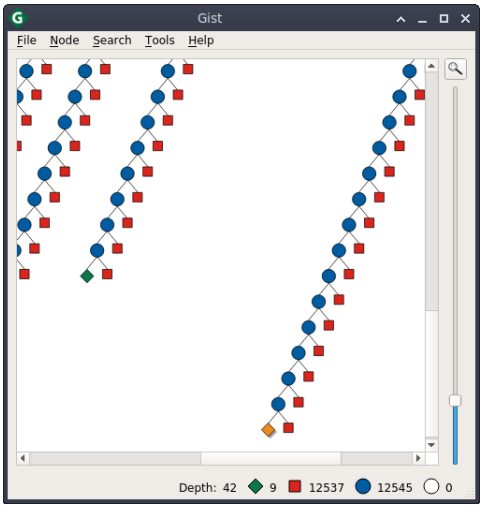


(a) Árbol de búsqueda

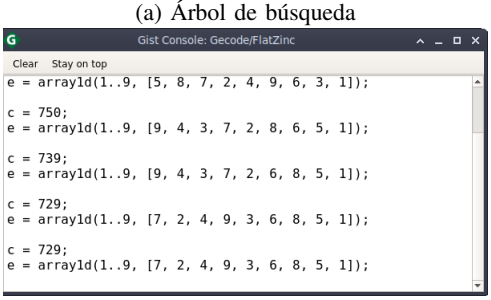


(b) Inspector de solución

Figura 6: Resultados para ejemplo 4



(a) Árbol de búsqueda



(b) Inspector de solución

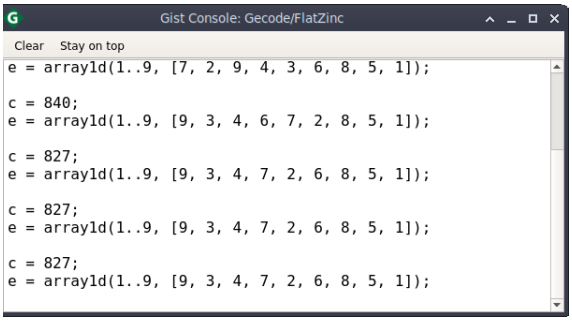
Figura 7: Resultados para ejemplo 5

IV-B4. Datos de entrada: Ejemplo 5:

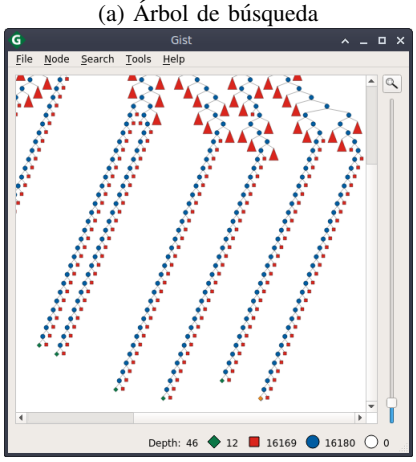
```
ACTORES = {Actor1, Actor2, Actor3,
           Actor4, Actor5, Actor6};

Escenas = [| 0,1,0,0,0,0,1,0,0,10
           | 0,1,1,1,0,1,1,1,1,20
           | 1,1,1,1,0,1,0,0,1,5
           | 0,1,1,1,0,0,1,1,1,5
           | 1,0,0,0,1,1,0,0,0,15
           | 0,0,0,0,1,0,0,1,0,19|];


Duracion = [2,1,1,1,3,4,2,3,1];
```

IV-B5. Datos de entrada: Ejemplo 6:

```
ACTORES = {Actor1, Actor2, Actor3,
           Actor4, Actor5, Actor6, Actor7};

Escenas = [| 0,1,0,0,0,0,1,0,0,10
           | 0,1,1,1,0,1,1,1,1,20
           | 1,1,1,1,0,1,0,0,1,5
           | 0,1,1,1,0,0,1,1,1,5
           | 1,0,0,0,1,1,0,0,0,15
           | 0,0,0,0,1,0,0,1,0,19
           | 0,1,0,0,1,0,0,0,0,8|];


Duracion = [2,1,1,1,3,4,2,3,1];
```



(a) Árbol de búsqueda



(b) Inspector de solución

Figura 8: Resultados para ejemplo 6

V. Segunda formalización del problema

Se toma toda la formalización hecha para el primer modelo pero con la siguientes restricciones que buscan romper las simetrías en el problema, puesto que existen permutaciones que se pueden asimilar como hacer el una operación de *reverse* en un arreglo. Así pues una forma de reducir el espacio de búsqueda es (basados en [2] [3]).

V-A. *Definición de funciones*

- SolCostActor$(K) \equiv$ Regresa el costo total de grabar la novela del actor K en Sol.
- GetCostActor$(K) \equiv$ Regresa el costo total de todas las escenas del actor K donde esta presente en Grabación$_{(e,K)}$

V-B. *Restricciones adicionales*

Nota: *Para este modelo se quita la restricción 3 dado que esta se encuentra implícitamente en la restricción 2*

Definición 1: Orden lexicográfico: Sea Σ un alfabeto dotado de un orden "$\leq$", para el presente modelo, para el alfabeto ASCII consideramos el orden usual. Extendemos el orden "$\leq$" de Σ a un orden " $\leq_{lg}$" de Σ^* mediante la definición siguiente:
Para cualesquiera tres palabras $\tau_0, \tau_1, \tau_2 \in \Sigma^*$ y cualesquiera símbolos $s_1, s_2 \in \Sigma$ rige la implicación

$$\left.\begin{array}{rl} \sigma_1 = & \tau_0 s_1 \tau_1 \\ \sigma_2 = & \tau_0 s_2 \tau_2 \end{array}\right\} \Longrightarrow s_1 < s_2$$

1. Sea C el conjunto solución del COP, se puede decir que

   $$\text{Sol} \in C, \exists \rho(.) | \rho(\text{Sol}) \in C$$

   (Se cumple la anterior propiedad para esta clase de problemas combinatorios, donde $\rho(.)$ es una función tal que devuelve la permutación de la secuencia).
   Así pues haciendo uso de la ***definición 1*** se puede establecer un orden lexicográfico para evitar las simetrías producidas por la soluciones en orden "inverso" de la solución, esto pues supone un orden establecido y así se evita que el espacio de búsqueda sea mas grande[3].
2. $$\forall\, a \in (\text{A} - \{\beta\}) : \text{UperBound}(a) - \text{LoweBound}(a) + 1 \geq \sum_{e \in E} \text{Grabación}_{(e,a)} \wedge \text{UperBound}(a) - \text{LowerBound}(a) + 1 \leq \kappa$$

   (Es decir con esta restricción lo que se busca es que todas las escenas, excepto las de β, su rango de posiciones posibles sea el que permita grabar la mayor cantidad de escenas posibles por cada uno de los actores (medida con la distancia entre las posiciones de cada escena para cada actor) y al mismo tiempo no puede exceder, o puede ser igual, a κ. Donde β es el actor con el mayor costo posible, $\kappa = \text{Max}_{i \in A}(\text{UperBound}(i) - \text{LowerBound}(i) + 1)$ esto es las correspondientes distancias entre posiciones de escenas de cada actor i y de ellas se toma la mayor de ellas, por lo que κ representa un "borde").
3. $$\forall\, a \in A \ : \ \text{SolCostActor}(a) \geq \text{GetCostActor}(a) \wedge \text{SolCostActor}(a) \leq \sum_{i \in \text{Sol}} d_i$$

   (Es decir el costo para el actor a tiene que ser mínimo el costo de cada duración de las escenas

[3]Los detalles de la implementación se dan mejor por entendidos observando la documentación de `MiniZinc` buscando la función `lex_lesseq`

en las que participa por lo que cobra y máximo todas las duraciones disponibles por lo que cobra).

V-C. *Función objetivo*

Se toma exactamente la misma función objetivo de los anteriores modelos.

VI. Implementación del segundo modelo

VI-A. *Ejemplo Trivial*

VI-A1. Datos de entrada: Usando los mismos datos de entrada que se usaron en la primera implementación para el llamado ejemplo ”Trivial”.

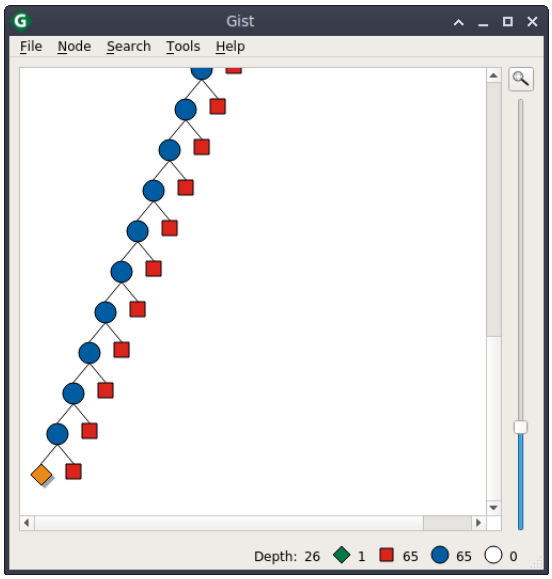


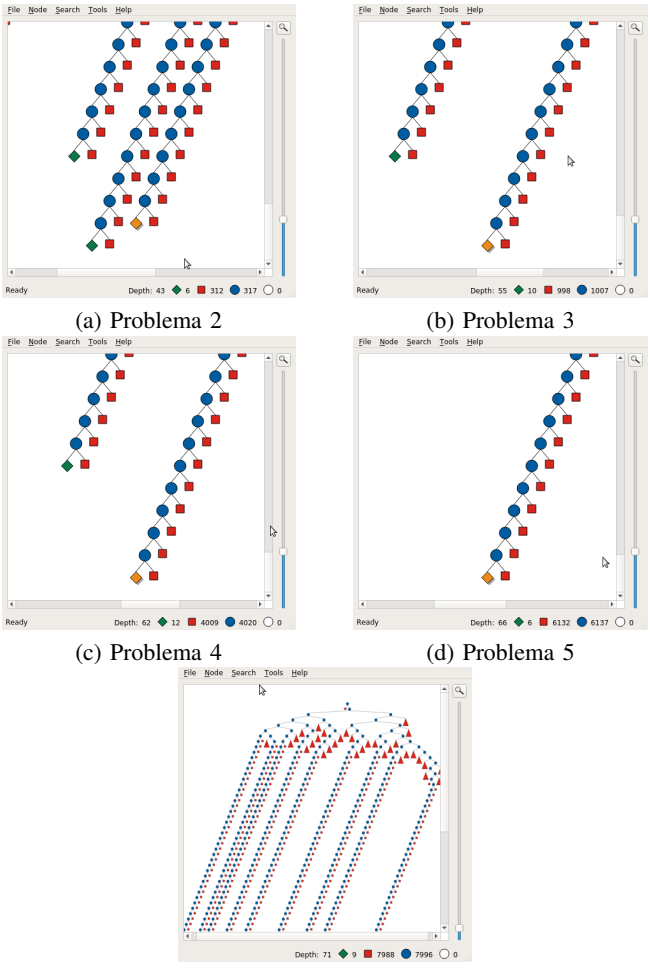
Figura 9: Árbol de búsqueda para el ejemplo Trivial

VI-A2. Análisis de resultados: En este ejemplo se ve una mejora respecto a la obtenida sin las restricciones adicionales, por un lado la búsqueda de la solución tiende a ”podarse” mas rápido en anchura (se redujo a casi la mitad para este caso particular), esto es por cuenta de la restricción impuesta para la eliminación de simetrías que corresponde a la restricción adicional 1. Sin embargo se observa que la profundidad del árbol es mayor porque el efecto colateral producido por calcular el borde de la restricción 2 es el de tener que realizar mas exploraciones hasta encontrar una solución para el (en ejemplos resultados posteriores se realizara una comparación para justificar el comportamiento producto de las restricciones adicionales redundantes 2 y 3).

VI-A3. Interpretación: Es exactamente la misma que la del modelo anterior, excepto por una variable de control que se tuvo en cuenta para determinar el impacto dentro de la solución, cabe aclarar que se uso la estrategia de distribución *por defecto* de `Minizinc`.

VI-B. *Otros ejemplos*

Se usaron exactamente los mismos datos de entrada usados para modelo 1, como también sus respectivas soluciones. Por lo que basta con mostrar los arboles de búsqueda y sus respectivos cambios. Nuevamente se



(a) Problema 2

(b) Problema 3

(c) Problema 4

(d) Problema 5

(e) Problema 6

Figura 10: Resultados arboles de búsqueda de problemas adicionales

puede observar, al igual que con trivial, que la cantidad de exploraciones en anchura se reduce casi a la mitad (estos valores se ”podan”) aunque supone una mayor profundidad en el árbol de búsqueda, sin embargo las restricciones redundantes permiten que para problemas con entradas mucho mas grandes se eviten arboles con mucha mas profundidad, es decir sí solo estuviera el orden lexicográfico junto con las demás se harían muchas mas exploraciones con estados fallidos en ”profundidad” y que no dirigen hacía la disminución del objetivo como claramente se puede observar en los 6 ejemplos, esto en contraste con lo observado para ”desenfreno” que

para el presente trabajo fue el ejemplo que se uso como *Benchmark*[4] de los demás y se realizarán las respectivas comparativas en una posterior sección.

VI-C. Desenfreno de pasiones

VI-C1. Datos de entrada: Se adjunta como imagen dado el tamaño.

```
ACTORES = {Actor1, Actor2, Actor3, Actor4, Actor5, Actor6, Actor7, Actor8} ;

Escenas = [| 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 10
           | 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 4
           | 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 5
           | 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 5
           | 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 5
           | 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 40
           | 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 4
           | 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 20|];

Duracion = [ 2, 1, 1, 1, 1, 3, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1];
```

Figura 11: Datos de entrada usados en desenfreno

VI-C2. Análisis de resultados: [5] Para este ejemplo se hizo uso de `Chuffed` y por lo tanto no se dibujo el árbol de la solución, pero los mismos criterios se aplicarán.

```
Output
----------
[ 5 more solutions ]
[4, 1, 10, 11, 13, 3, 12, 2, 6, 8, 9, 7, 20, 5, 15, 14, 17, 18, 16, 19]
Costo: 871
Border: 16
---------
==========
Finished in 4h 12m 28s
```

Figura 12: Resultado con Chuffed

Como se nombro en el anteriores ejemplos, sucede que a medida que crece el tamaño, sumado a que en este ejemplo hay mas participación en escenas, que muchas escenas son grabadas por un grupo diferente de actores (las columnas de la matriz de entrada son distintas una de otra), en este punto es cuando las restricciones de ”borde” y redundantes toman mayor protagonismo.

VII. Análisis experimental

En esta sección, en adición con las de los análisis hechos para cada modelo (en los que se expuso una serie de ejemplos), se exponen los tiempos de ejecución, las comparaciones entre cada modelo y solver, en un primer acercamiento con busqueda por defecto de `MiniZinc`.

VII-A. Comparaciones

[4]El termino *Benchmark* se refiere, según el diccionario Merriam-Webster, a: Es un problema o test estandarizado que sirve como base de evaluación o comparación.

[5]Ejecuciones realizadas en un computador con procesador Core i5 7ma Generación y 8 Gb de RAM

Cuadro III: Tabla de comparación entre Solvers y problemas para Modelo 1

Ejemplos	Geocode				Chuffed			
	Sol	Fails	Expa	Depth	Sol	Fail	Expa	Depth
Trivial	1	106	106	15	2	64	137	33
2	9	492	500	32	8	365	955	62
3	15	1656	1670	38	6	651	8278	77
4	20	8259	8278	43	7	1581	2585	92
5	9	12537	12545	42	13	1814	4017	99
6	12	16169	16180	46	16	4027	7185	110

Cuadro IV: Tabla de comparación entre Solvers y problemas para Modelo 1 versión 2

Ejemplos	Geocode				Chuffed			
	Sol	Fails	Expa	Depth	Sol	Fail	Expa	Depth
Trivial	1	65	65	26	3	42	205	54
2	6	312	317	43	5	132	530	89
3	10	998	1007	55	8	347	1259	99
4	12	4009	4020	62	13	704	2200	111
5	6	6132	6137	66	3	547	1200	129
6	9	7988	7996	71	18	2231	5650	154

Se puede notar en los resultados anteriores que estos varían significativamente dependiendo del tipo de solver que se escoja para la ejecución del modelo final. Teniendo en cuenta que cada instancia es en cierta medida mas compleja o grande que la anterior, podemos apreciar que para las primeras 3 instancias obtenemos resultados mas favorables por parte de Gecode, por el contrario con las siguientes 3 instancias los resultados favorecen el uso de Chuffed. Cabe aclarar que ”favorable”significa que se expanden menos nodos y se falla menos.

Debido al comportamiento antes mencionado entonces, a medida que la entrada sea mas compleja, con Chuffed se puede llegar a un resultado óptimo mucho mas rápido, y una prueba de ello, es el resultado de la instancia ”Desenfreno”.

VII-B. Gráficos tiempos de ejecución

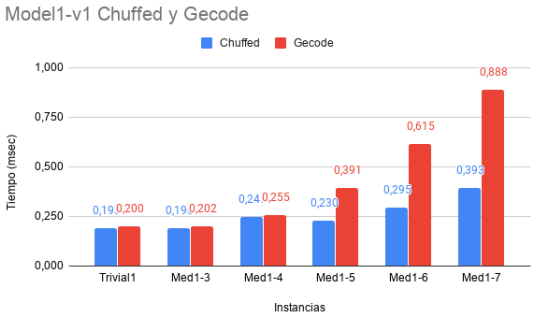


Figura 13: Modelo 1 - Chuffed vs Geocode

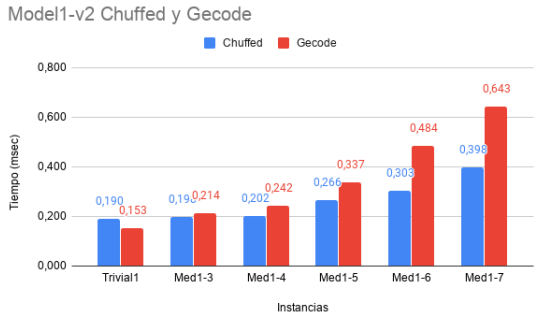


Figura 14: Modelo 1 V2 - Chuffed vs Geocode

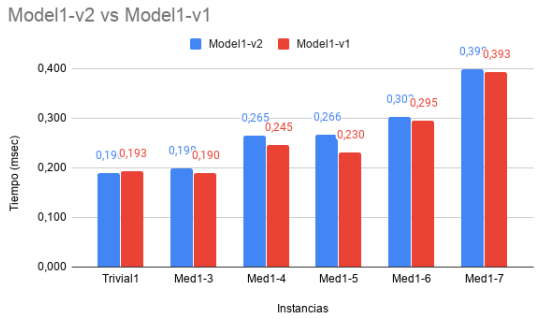


Figura 15: Comparación entre modelos

Según los resultados obtenidos por las instancias medianas y ”Desenfreno”se puede concluir que el Modelo 1-v2 resulta mas ventajoso y apropiado para manejar cualquier tipo de entrada, especialmente las grandes.

VIII. Estrategias de búsqueda

Para las instancias de prueba antes desarrolladas sin incluir Desenfreno, Para el Modelo 1-v2, con el solver Gecode, una estrategia de búsqueda que resulta muy efectiva es la que tiene en cuenta la anotación de elección de variables mediante Domwdeg[6] y restringiendo las variables con IndomainSplit[7] sobre Sol.

Se evidencia que para estas instancias la expansión de nodos y los fallos encontrados son considerablemente menores en comparación al modelo sin utilizar ninguna estrategia de búsqueda. Esto se puede intuir ya que la elección de variables y la forma como se restringen, para este problema resulta beneficioso en el sentido de que se esta teniendo en cuenta sobre la ejecución,

[6]Se elije la variable con el valor más pequeño del tamaño del dominio dividido por el grado ponderado, que es el número de veces que ha estado en una restricción que causó fallas anteriormente en la búsqueda.

[7]bisecciona el dominio de variables excluyendo la mitad superior.

Cuadro V: Resultados con la estrategia de búsqueda

Ejemplos	Geocode			
	Sol	Fails	Expa	Depth
Trivial	1	43	43	31
2	8	275	282	42
3	17	840	856	53
4	22	4099	4120	59
5	13	2761	2773	70
6	13	4961	4973	79

el comportamiento de las variables en función de su dominio y de sus fallas. También cabe recalcar que para la implementación de esta estrategia únicamente el solver de Gecode es capaz de hacer uso de esta. Para el caso de Desenfreno no se pudo establecer oportunamente si este modelo obtiene un resultado mas rápido que el anterior expuesto.

IX. Modelo 2 del problema

Para esta segunda parte se añaden nuevas restricciones sobre la grabación de la novela.

1. Disponibilidad de los actores: en general todos los actores deben estar disponibles todo el tiempo, pero hay cierto actores que por ser estrellas pueden imponer restricciones como por ejemplo un máximo numero de unidades de tiempo en el estudio. Por ejemplo Narciso Perfiles podría decir 10, y el actor 3 podría decir 15.
2. Puede haber actores que no la van bien con otros actores y sería ideal que el tiempo que estuvieran en el estudio al mismo tiempo fuera el menor posible (pero tenga en cuenta que la prioridad es reducir costo).

```
ACTORES = {Actor1, Actor2, Actor3, Actor4, Actor5, Actor6, Actor7, Actor8} ;

Escenas=[| 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 10 %Actor 1
         | 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 4  %Actor 2
         | 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 5  %Actor 3
         | 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 5  %Actor 4
         | 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 5  %Actor 5
         | 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 40 %Actor 6
         | 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 4  %Actor 7
         | 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 20|]; %Actor8

Duracion = [2 , 1 , 1, 1, 1, 3, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1];

Disponibilidad =[|Actor1, 0
                 |Actor2, 0
                 |Actor3, 23
                 |Actor4, 0
                 |Actor5, 80
                 |Actor6, 10
                 |Actor7, 5
                 |Actor8, 100|];

Evitar =[|Actor1, Actor3
         |Actor1, Actor4
         |Actor2, Actor3
         |Actor2, Actor4|];
```

Figura 16: Modelo 2 con Disponibilidad y evitar

- “Disponibilidad” especifica el nombre del actor seguido por la restricción de tiempo (0 quiere

decir tiempo ilimitado). Esta estructura contiene una entrada por cada uno de los actores.
- "Evitar" especifica los actores que le gustaría evitar encontrarse en las grabación. Esta estructura puede contener un numero de restricciones variable.

X. Formalización del problema

Sea E el conjunto de todas las escenas que se desean grabar para la novela tal que $e \in E$ y sea A el conjunto de actores que hay disponibles para grabar la novela tal que $a \in A$.

X-A. Parámetros

- Grabación$_{(e,a)}$ representa sí el actor a graba la escena e (se representa así: $1 \equiv$ Sí graba y $0 \equiv$ No graba)
- d_e representa la duración de la escena e.
- Costo$_a$ representa el costo por unidad de tiempo para el actor a.
- Disponibilidad$_a$ representa la cantidad de unidades de tiempo disponible que tienen el actor a para grabar (donde $0 \equiv$ Tiene tiempo ilimitado para grabar y mayor que $0 \equiv$ tiene cierta cantidad de tiempo disponible).
- Evitar$_a$ representa que un actor a desea estar lo mas lejos posible de un cierto conjunto de actores Ψ en toda la grabación, es decir la participación de sus escenas tiene que ser, en lo posible, separadas de ellos. (Por lo que Evitar$_a \in M$, $M \subset \mathcal{P}(A - \{a\})$).

X-B. Variables

- Sol$_i$ representa el orden de grabación i para cierta escena e, donde $i \in [1..\text{Card}(E)]$.

X-C. Dominio

- Sol$_i \in E$

X-D. Definición de Funciones

- UperBound$(K) \equiv$ Regresa el índice i en Sol$_i$ de la última escena del actor K
- LowerBound$(K) \equiv$ Regresa el índice i en Sol$_i$ de la primera escena del actor K
- SolCostActor$(K) \equiv$ Regresa el costo total de grabar la novela del actor K en Sol.
- GetCostActor$(K) \equiv$ Regresa el costo total de todas las escenas del actor K donde esta presente en Grabación$_{(e,K)}$

X-E. Definición de Predicados

Nota: *Se usa la abreviación Disp$_a$ con Disponbilidad$_a$ para poder ajustar el tamaño en el lado derecho del predicado.*

- $\text{Disp}(K) \equiv$

$$\begin{cases} \text{true} & \text{Sí } \text{Disp}_K = 0 \\ \sum_{q \in P(K)} d_q \leq \text{Disp}_K & \text{Sí } \text{Disp}_K > 0 \end{cases}$$

(Donde $P(.)$ es una función tal que retorna el intervalo que representa el mapeo del correspondiente e de "." al momento desde donde inicia la grabación de su primera escena hasta la última que le toca en Sol$_i$).

- $\text{Evit}(K, M) \equiv$

$$\text{LowerBound}(K) \leq \text{UperBound}(M)$$
$$\vee\, \text{LowerBound}(M) \leq \text{UperBound}(K)$$

(Es decir que el actor K requiere la grabación lo mas lejos posible de la finalización de la última escena de M y vice versa).

X-F. Restricciones

1. $\{\text{Sol}_i \neq \text{Sol}_j \,|\, \forall i, j \in [1..\text{Card}(E)] \wedge i \neq j\}$ (Es decir no se pueden grabar dos escenas en un mismo espacio de tiempo, es decir el mismo día).
2. $$\text{UperBound}(\beta) - \text{LowerBound}(\beta) + 1 \quad = \quad \sum_{e \in E} \text{Grabación}_{(e,\beta)}$$

   (Es decir la duración total debe ser tal que se grabe la mayor cantidad de escenas sucesivas posibles para ese actor, deben de quedar una seguida de la otra. Donde β corresponde al actor con el mayor costo posible en toda la grabación).
3. Sea C el conjunto solución del COP, se puede decir que
   $$\text{Sol} \in C,\ \exists \rho(.) | \rho(\text{Sol}) \in C$$
   (Se cumple la anterior propiedad para esta clase de problemas combinatorios, donde $\rho(.)$ es una función tal que devuelve la permutación de la secuencia).
   Así pues haciendo uso de la ***definición 1*** se puede establecer un orden lexicográfico para evitar las simetrías producidas por la soluciones en orden "inverso" de la solución, esto pues supone un orden establecido y así se evita que el espacio de búsqueda sea mas grande.

4.

$$\forall a \in (\text{A} - \{\beta\}) : \text{UperBound}(a) - \text{LoweBound}(a) + 1 \geq \sum_{e \in E} \text{Grabación}_{(e,a)} \wedge \text{UperBound}(a) - \text{LowerBound}(a) + 1 \leq \kappa$$

(Es decir con esta restricción lo que se busca es que todas las escenas, excepto las de β, su rango de posiciones posibles sea el que permita grabar la mayor cantidad de escenas posibles por cada uno de los actores (medida con la distancia entre las posiciones de cada escena para cada actor) y al mismo tiempo no puede exceder, o puede ser igual, a κ. Donde β es el actor con el mayor costo posible, $\kappa = \text{Max}_{i \in A}(\text{UperBound}(i) - \text{LowerBound}(i) + 1)$ esto es las correspondientes distancias entre posiciones de escenas de cada actor i y de ellas se toma la mayor de ellas, por lo que κ representa un "borde").

5.

$$\forall a \in A \ : \ \text{SolCostActor}(a) \geq \text{GetCostActor}(a) \wedge \text{SolCostActor}(a) \leq \sum_{i \in \text{Sol}} d_i$$

(Es decir el costo para el actor a tiene que ser mínimo el costo de cada duración de las escenas en las que participa por lo que cobra y máximo todas las duraciones disponibles por lo que cobra).

6. $\forall a \in A \ : \text{Disp}(a)$ (Es decir se debe de cumplir para todos los actores el predicado).
7. $\forall a \in A, \forall a' \in \text{Evitar}_a \ : \text{Evit}(a, a')$ (Lo mismo que el anterior predicado).

X-G. *Función Objetivo*

Minimizar:

$$\sum_{a \in A} \sum_{k \in P(a)} d_k \times \text{Costo}_a$$

(De igual forma P se define como se uso en uno de los predicados).

Sujeto a: Que el tiempo que estuvieran en el estudio al mismo tiempo fuera el menor posible.

XI. Implementación del modelo 2 del problema

XI-A. *Ejemplo trivial*

XI-A1. *Datos de entrada:*

```
ACTORES = {Actor1, Actor2,
           Actor3, Actor4};

Escenas = [| 1,1,0,0,0,0,10
           | 0,1,1,1,0,1,20
           | 0,0,0,0,1,1,15
           | 1,1,0,0,1,1,13|];

Duracion = [2,1,1,1,3,4];

Disponibilidad =[|Actor1, 5
                 |Actor2, 8
                 |Actor3, 0
                 |Actor4, 10|];

Evitar =[|Actor1, Actor2
         |Actor2, Actor3|];
```

XI-A2. *Análisis de resultados:*

Aunque vale hacer la aclaración de que aún cuando se conservaron todas las restricciones y parámetros del modelo 1 V2 del problema, este modelo no es directamente comparable (porque son distintos en realidad) pero sí se hace un análisis resulta *per se* la evidencia de que las nuevas restricciones ayudan a restringir aún más el dominio de las variables implicadas en el problema, por lo que se espera que se realicen menos exploraciones (se poda mucho mas), aunque puede crecer, nuevamente, la profundidad del árbol.

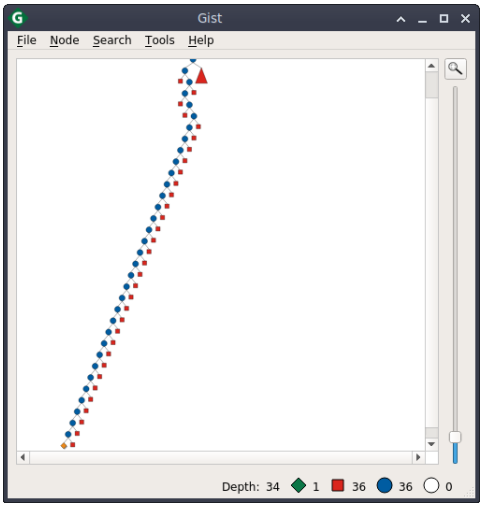


Figura 17: Árbol de búsqueda para el ejemplo Trivial

XI-A3. *Interpretación de resultados:*

En este caso a parte de el orden de las escenas y el coste de las escenas, también nos interesa conocer la suma total de los tiempos que se ha compartido para cada par de actores que desean evitarse.

```
Orden Escenas: [4, 3, 6, 2, 1, 5];
```

```
Costo: 450;
Tiempo Compartido: 6;
```

Cuadro VI: Tabla inicial de escenas para Trivial

Escenas	1	2	3	4	5	6	Costo
Actor 1	1	1	0	0	0	0	10
Actor 2	0	1	1	1	0	1	20
Actor 3	0	0	0	0	1	1	15
Actor 4	1	1	0	0	1	1	13

En tabla se puede observar el cambio en el orden de las escenas pero con la particularidad de que se guardan las distancias correspondientes entre los actores. Se puede

Cuadro VII: Tabla resultado de escenas para Trivial

Escenas	4	3	6	2	1	5	Costo
Actor 1	0	0	0	1	1	0	10
Actor 2	1	1	1	1	0	0	20
Actor 3	0	0	1	0	0	1	15
Actor 4	0	0	1	1	1	1	13

observar la distancia entre los actores 1 y 2, el cual, para este ejemplo, se encontró que sí el actor 2 inicia primero la escena 4 y de ahí adelante hasta la escena 2 solo se lo encuentra allí. En cambio el actor 3 no goza de la misma ”fortuna” sino que desde la escena 6 se lo encuentra dado que la tienen que grabar ese día (es inevitable) por lo que lo que se busca es que la terminación de alguno de los dos sea antes de la del otro, que es lo que se observa para el actor 2. Esta prueba fue realizada usando la búsqueda *por defecto* de `MiniZinc`.

XI-B. Otros ejemplos

XI-B1. Datos de entrada: Ejemplo 2:

```
ACTORES = {Actor1, Actor2,
    Actor3, Actor4} ;
Escenas = [| 0,1,0,0,0,0,10
           | 0,1,1,1,0,1,20
           | 1,1,1,1,0,1,5
           | 1,0,0,0,1,1,15|];
Duracion = [2,1,1,1,3,4];
Disponibilidad =[|Actor1, 5
                 |Actor2, 8
                 |Actor3, 0
                 |Actor4, 10|];
Evitar =[|Actor1, Actor2
         |Actor2, Actor3|];
```

XI-B2. Datos de entrada: Ejemplo 3:

```
ACTORES = {Actor1, Actor2,
    Actor3, Actor4} ;
Escenas = [| 0,1,0,0,0,0,1,10
           | 0,1,1,1,0,1,1,20
           | 1,1,1,1,0,1,0,5
           | 1,0,0,0,1,1,0,15|];
Duracion = [2,1,1,1,3,4,2];
Disponibilidad =[|Actor1, 8
                 |Actor2, 9
                 |Actor3, 0
                 |Actor4, 10|];
Evitar =[|Actor1, Actor2
         |Actor2, Actor3|];
```

XI-B3. Datos de entrada: Ejemplo 4:

```
ACTORES = {Actor1, Actor2, Actor3,
                Actor4, Actor5} ;
Escenas = [| 0,1,0,0,0,0,1,0,0,10
           | 0,1,1,1,0,1,1,1,1,20
           | 1,1,1,1,0,1,0,0,1,5
           | 0,1,1,1,0,0,1,1,1,5
           | 1,0,0,0,1,1,0,0,0,15|];
Duracion = [2,1,1,1,3,4,2,3,1];
Disponibilidad =[|Actor1, 13
                 |Actor2, 14
                 |Actor3, 15
                 |Actor4, 14
                 |Actor5, 10|];
Evitar =[|Actor1, Actor2
         |Actor2, Actor3
         |Actor4, Actor5|];
```

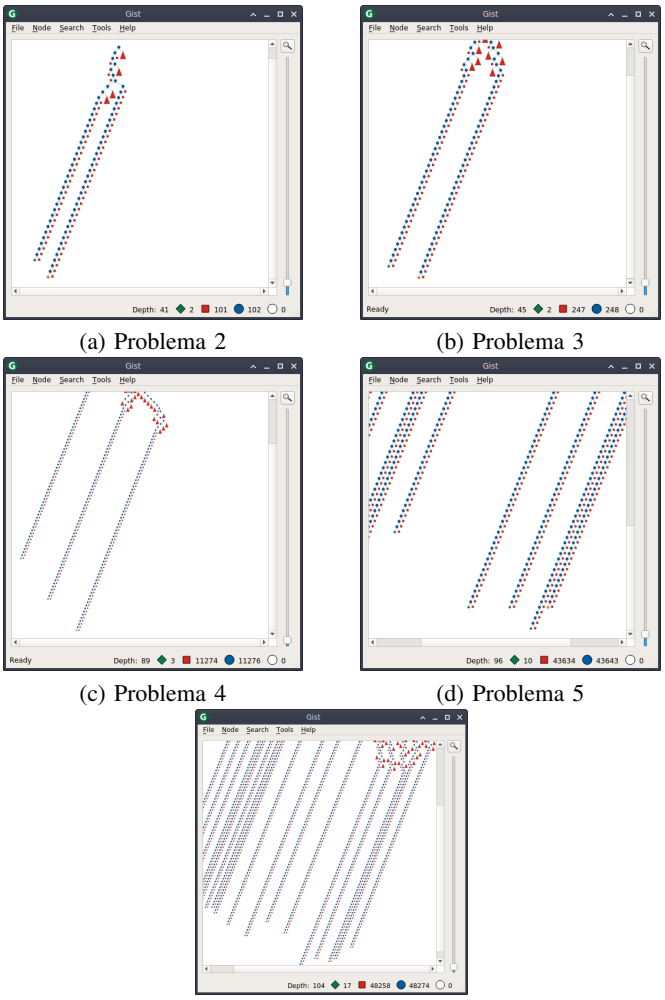
XI-B4. Datos de entrada: Ejemplo 5:

```
ACTORES = {Actor1, Actor2, Actor3,
    Actor4, Actor5, Actor6} ;
Escenas = [| 0,1,0,0,0,0,1,0,0,10
           | 0,1,1,1,0,1,1,1,1,20
           | 1,1,1,1,0,1,0,0,1,5
           | 0,1,1,1,0,0,1,1,1,5
           | 1,0,0,0,1,1,0,0,0,15
           | 0,0,0,0,1,0,0,1,0,19|];
Duracion = [2,1,1,1,3,4,2,3,1];
Disponibilidad =[|Actor1, 15
                 |Actor2, 15
                 |Actor3, 16
                 |Actor4, 17
                 |Actor5, 12
                 |Actor6, 0|];
```

XI-B5. Datos de entrada: Ejemplo 6:

```
ACTORES = {Actor1, Actor2, Actor3,
    Actor4, Actor5, Actor6, Actor7};
Escenas = [| 0,1,0,0,0,0,1,0,0,10
           | 0,1,1,1,0,1,1,1,1,20
           | 1,1,1,1,0,1,0,0,1,5
           | 0,1,1,1,0,0,1,1,1,5
           | 1,0,0,0,1,1,0,0,0,15
           | 0,0,0,0,1,0,0,1,0,19
           | 0,1,0,0,1,0,0,0,0,8|];
Duracion = [2,1,1,1,3,4,2,3,1];
Disponibilidad =[|Actor1, 15
                 |Actor2, 15
                 |Actor3, 16
                 |Actor4, 17
                 |Actor5, 12
                 |Actor6, 0
                 |Actor7, 0|];
Evitar =[|Actor1, Actor2
         |Actor2, Actor3
         |Actor4, Actor5
         |Actor1, Actor6|];
```

XI-B6. Resultados: Los resultados fueron



(a) Problema 2

(b) Problema 3

(c) Problema 4

(d) Problema 5

(e) Problema 6

Figura 18: Resultados arboles de búsqueda de problemas adicionales

```
Running MOdelPart2.mzn
[4, 3, 6, 2, 1, 5];450;6;
----------
==========
Finished in 637msec
```

(a) Problema 2

```
Running MOdelPart2.mzn
[2, 7, 4, 3, 6, 1, 5];400;12;
----------
[5, 1, 6, 3, 4, 2, 7];390;10;
----------
==========
Finished in 149msec
```

(b) Problema 3

```
----------
[5, 1, 6, 2, 7, 9, 4, 3, 8];530;13;
----------
[5, 1, 6, 9, 4, 3, 2, 7, 8];520;11;
----------
==========
Finished in 674msec
```

(c) Problema 4

```
----------
[1, 5, 8, 6, 3, 4, 2, 9, 7];739;22;
----------
[1, 5, 8, 6, 4, 9, 3, 2, 7];729;21;
----------
==========
Finished in 2s 293msec
```

(d) Problema 5

```
[1, 5, 8, 6, 4, 3, 2, 9, 7];843;22;
----------
[1, 5, 8, 6, 4, 9, 3, 2, 7];841;21;
----------
==========
Finished in 3s 28msec
```

(e) Problema 6

Figura 19: Soluciones para cada ejemplo

XI-C. Desenfreno de pasiones

XI-C1. Datos de entrada: Se usarán los mostrados en el del enunciado del problema (fig. 16).

XI-C2. Análisis de resultados: Aunque no son directamente comparables con el modelo 1, se puede observar como con un dominio mas restringido permite acortar el espacio de búsqueda, otra observación que se puede hacer es que el costo aumenta un poco porque las restricciones de disponibilidad y evitar hacen que ciertas posiciones que son mas ”óptimas”no se puedan tomar. Aún así la prioridad sigue siendo minimizar el costo, independientemente de que no puedan evitarse, sin embargo se ve que que el tiempo compartido (unidades de tiempo) aumenta a medida que se añaden mas restricciones de “evitar”.

```
Output
[ 13 more solutions ]
[4, 1, 11, 10, 13, 12, 2, 3, 9, 6, 8, 7, 5, 15, 14, 16, 17, 18, 20, 19]
Costo: 880
Tiempo: 37
----------
==========
Finished in 7m 37s
```

Figura 20: Resultado con Chuffed

XII. Análisis experimental

De igual manera en esta sección se abordara los tiempos de ejecución de cada Solver.

Cuadro VIII: Tabla de comparación entre Solvers y problemas para Modelo 2

Ejemplos	Geocode				Chuffed			
	Sol	Fails	Expa	Depth	Sol	Fail	Expa	Depth
Trivial2	1	36	36	34	1	29	78	41
2	2	102	102	41	2	27	153	70
3	2	247	248	45	4	65	423	104
4	3	11274	11276	89	5	117	754	144
5	10	43722	43731	96	5	188	1001	161
6	17	48205	48221	104	10	614	2415	178

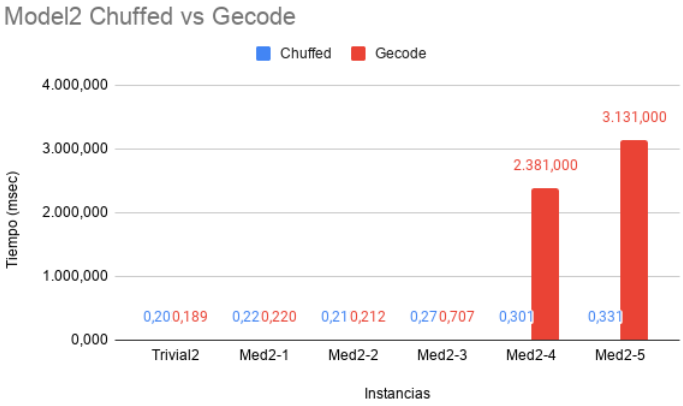


Figura 21: Comparación Chuffed vs Geocode

Nuevamente se puede apreciar que los resultados tanto en expansión de nodos, fallas y tiempos, Chuffed se presenta como el solver indicado para la ejecución de las instancias de una forma mas eficaz, según claro, este problema en particular.

XIII. Estrategias de búsqueda

Cuadro IX: Resultados con estrategia de búsqueda

Ejemplos	Geocode			
	Sol	Fails	Expa	Depth
Trivial2	1	39	39	32
2	3	110	112	39
3	6	249	254	47
4	8	2620	2627	81
5	17	1971	1987	88
6	16	2739	2754	94

En este apartado se busco una estrategia de distribución que fuera mejor al trabajo que hiciera *búsqueda por defecto* de `MiniZinc`, por lo que se probaron

varias búsquedas enteras para escoger la variable y su respectivo valor. Para el caso del *Costo total* se uso escoger siempre el orden de entrada, esto es que cuando se minimiza una variable según la implementación que se haya hecho se puede alcanzar el valor mas rápido si se construye un arreglo de valores y siempre se escoge el primero de tal forma que haga una búsqueda entera siempre con los valores menores, por lo que probar otras estrategias no mejoro la presente estrategia expuesta. Luego esta la imposición sobre Sol, en este caso se probaron *First fail* que toma la variable que tiene el dominio mas pequeño y se le toma con *Valor mas pequeño del dominio* resulto en una mejoría notable respecto de la búsqueda por defecto, se observo también que usando *escoger la variable con el valor mas pequeño de todos los dominios* hubo otra gran mejora, sin embargo uno de los candidatos que mejor comportamiento tuvo fue *escoger la variable que tiene un cierto peso mayor* que consiste, a groso modo, a que se escoja la variable que dado el conjunto de restricciones haya aportado menos estados fallidos antes. Finalmente la estrategia que mejor comportamiento obtuvo en general (fig. IX vs fig. VIII) fue el de escoger la variable con pesos y escoger el valor de esta variable con *bisecciona el dominio de variables excluyendo la mitad superior*.

XIV. Conclusión

El orden de las escenas es un problema realmente interesante que plantea todo un desafío contra la complejidad que añade el tener que permutar, en el caso del modelo mas trivial con un problema realmente complicado, hasta $n!$ posibles ordenes de escenas a grabar. Aunque este problema en el ámbito de los *COP* no es algo ajeno puesto que en la literatura se puede encontrar problemas similares como el *Scene allocation*, este en cambio supone una mayor libertad en el dominio como también un planteamiento un poco distinto a como se desarrollan al anteriormente nombrado, haciendo mas interesante de resolver este problema y sus particularidades. Es un problema que requiere ser estudiado mas a fondo por su naturaleza combinatoria, por lo que profundizar en esta rama y poder encontrar alguna propiedad o teorema que se pueda aplicar a este problema resultaría, junto con las otras restricciones, en la resolución del problema de forma mas eficiente.

Referencias

[1] K. Apt, *Principles of Constraint Programming*. USA: Cambridge University Press, 2003.

[2] "Combinatorial optimisation and constraint programming 2019, topic 5: Symmetry," Sep 2019. [Online]. Available: http://user.it.uu.se/ pierref/courses/COCP/slides/T05-Symmetry.pdf

[3] N. Baxter, G. Chu, and P. J. Stuckey, "Symmetry declarations for minizinc," in *Proceedings of the Australasian Computer Science Week Multiconference*, ser. ACSW '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2843043.2843058