Programación con Restricciones **(Entrega: 05/05/2020)**

Informe del Proyecto

Instructor: R. Duque, J.F. Díaz *Name:* Juan David Díaz, *Cód.:* 1905397

Introducción

La programación con restricciones es un campo muy explorado y con una fundamentación formal que permite resolver problemas cuya naturaleza está dada por distintas restricciones y dominios.

Este proyecto resuelve un problema de **Scheduling**, el cual consiste en determinar en qué orden debe realizarse un conjunto de eventos, en este caso particular, para minimizar el costo del rodaje de una novela.

En este documento se describirá el problema y la forma en que fue resuelto, y cómo se integró a una aplicación.

Descripción del Problema

Este problema consta de una secuencia de escenas con una duración dada que pueden ser grabadas en cualquier orden. Ciertos actores participan en la grabación de cada escena, y cada actor cobra un valor por unidad de tiempo.

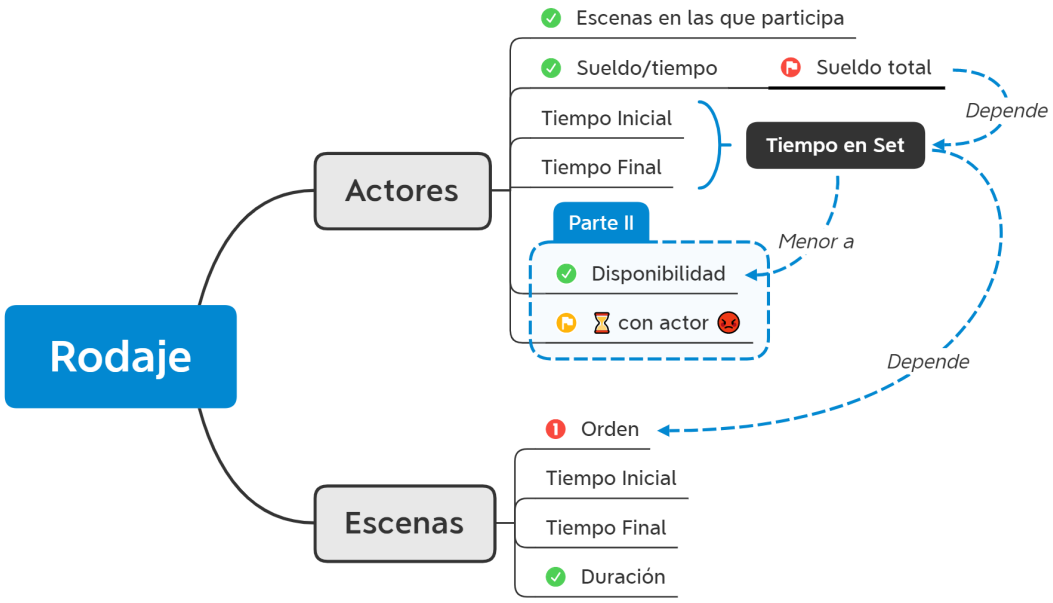

Figure 1: Problema

Los actores estarán en el set desde la primera escena que participan hasta la última, por lo que hay que pagarles así estén en el set durante escenas en las que no participan, por lo que lo ideal es

encontrar un orden para las escenas de forma que se minimice el costo de los actores. Es evidente que los actores que cobran más por unidad de tiempo deben permanecer el menor tiempo posible en el set, por lo que el orden de las escenas es menos sensible respecto a los actores que cobran menos por unidad de tiempo.

Para la parte dos del problema se introducen nuevas variables al problema, como la disponibilidad de tiempo de los actores, que restringe el tiempo total que puede pasar un actor en el set; y la posibilidad de que un actor no se lleve bien con otro, por lo que es ideal minimizar el tiempo que pasan juntos durante el rodaje.

La figura 1 muestra de forma general cuáles son los parámetros y variables del problema y cómo se relacionan entre sí.

Restricciones. Parte I

n esta sección se describirán las restricciones implementadas en el modelo y cómo se llegó a la conclusión de que eran necesarias.

Se identifican como datos de entrada los actores, las escenas en las que participan, la duración de estas escenas, y lo que cobran los actores por unidad de tiempo.

Como variables del problema, el orden de las escenas, el tiempo de inicio y final de cada una, así como el tiempo de entrada y salida del set de los actores. El sueldo de cada actor también es una variable del problema.

Se consideró que era necesario incluir en el modelo un arreglo que representara el orden de las escenas pues es el dominio más importante del problema.

En el modelo no se usa demasiado este arreglo para generar otras restricciones, pero es bastante útil para entender la solución al problema.

Inicialmente se creó una restricción que intentaba relacionar los tiempos de inicio y final con el orden de las escenas mediante la siguiente restricción.

$$orden\_escenas_{i-1} = j \Leftrightarrow start\_time_i = end\_time_j$$

Esta restricción implica que la escena i va después de la escena j sí y solo sí la escena j termina en el momento en que la escena i inicia.

A pesar de que esta restricción parece funcionar, para ciertas instancias el solver tenía problemas para encontrar una solución.

$$arg\_sort(start\_time, orden\_escenas)$$

Finalmente, se encontró una restricción global de minizinc que resolvía este problema. Esta restricción implica que el arreglo que ordena los tiempos de inicio de las escenas era el orden de las escenas.

Ahora, es necesario asegurar que las escenas no ocurran al mismo tiempo, pues solo se puede grabar una escena a la vez.

$$start\_time_i + Duracion_i \leq start\_time_j \vee start\_time_j + Duracion_j \leq start\_time_i$$

Esta restricción indica que una escena solo puede empezar después de que haya pasado cierto tiempo (la duración de cada escena) de haber iniciado otra escena.

A pesar de que esta restricción funcionara bien, existía una restricción global que hacía lo mismo, y por lo general las restricciones globales son más eficientes, así que se decidió usarla.

$$disjunctive\_strict(start\_time, Duracion)$$

De acuerdo con la documentación, esta restricción indica que un conjunto de tareas con tiempo de inicio y duración no se pueden superponer en el tiempo. La restricción strict en particular indica que aunque el tiempo de una tarea sea despreciable (0), no puede ejecutarse en medio de otra tarea.

Teniendo ya el tiempo de inicio, se define que cada escena termina pasada su duración desde el tiempo de inicio.

$$end\_time_i = start\_time_i + Duracion_i$$

Para facilitar la implementación de otras restricciones, se almacenan las escenas en que participa cada actor en conjuntos, que a su vez se almacenarán en un arreglo.

Así, determinar el tiempo en que un actor entra al set y sale del set se hace más fácil.

$$actor\_end\_time_i = max(\{end\_time_j : j \in escenas\_actor_i\})$$

Pues un actor entra al set cuando inicia su escena más temprana, y sale del set cuando termina su escena más tardía.

$$actor\_start\_time_i = min(\{start\_time_j : j \in escenas\_actor_i\})$$

Teniendo esta información, calcular el tiempo que pasa un actor en el set es restar el tiempo en que se va del set con el tiempo en que entra al set. Y el costo de ese actor es este tiempo multiplicado por lo que cobra por unidad de tiempo.

Con las restricciones anteriores es suficiente para definir la primera parte del proyecto.

Ahora, la función a minimizar (función objetivo) es la sumatoria de todos los sueldos de los actores. Este es el costo total, en cuanto a actores se refiere, de rodar la novela

$$costo\_total = \sum_{i=1}^{n} sueldo\_actor_i$$

Restricciones. Parte II

La segunda parte añade dos datos más a la entrada: el tiempo máximo que puede pasar cada actor en el set, y una lista de parejas de actores que no se llevan bien.

Esto agrega dos restricciones más al problema original. La primera restringe el tiempo que pueden pasar los actores en el set. Como ya se tiene el tiempo que pasa cada actor en el set, implementar

esta restricción es más sencillo, pero hay una particularidad, y es que esta restricción no aplica a los actores cuya disponibilidad es cero en los datos de entrada, entonces la restricción es la siguiente:

$$Disponibilidad_{i,2} \neq 0 \implies tiempo\_en\_set_i \leq Disponibilidad_{i,2}$$

O sea que si la disponibilidad del actor es distinta de cero, se restringe el tiempo en el set del actor a ser menor o igual que la disponibilidad de ese actor.

Finalmente, el tiempo que pasan juntos una pareja de actores que tiene conflictos puede expresarse como el tiempo en que se va el que se va antes menos el tiempo en que llega el que llega al último.

$$f\_t(a) = min(actor\_end\_time_a) - max(actor\_start\_time_a) : a \in Evitar_i$$

Pero en el caso que uno de los actores se vaya antes de que el otro llegue, el tiempo que comparten será negativo, y esto no tiene sentido, por lo tanto, es necesario reescribir la expresión que totaliza el tiempo que comparten las parejas de actores con conflictos.

$$tiempo\_compartido = \sum_{i \in Evitar} f\_t\_f(i) \begin{cases} 0 & f\_t(i) < 0 \\ f\_t(i) & f\_t(i) \geq 0 \end{cases}$$

A pesar de haber resuelto este problema, se conserva una restricción que dice que el tiempo compartido debe ser mayor a cero, solo para podar el dominio de esta variable.

Entonces, la función objetivo debe incluir esta variable, pero recordando que es más importante el costo, se asigna al tiempo compartido menos peso, en este caso con una proporción de dos a uno.

$$f(orden\_escenas) = 2 \cdot costo\_total + tiempo\_compartido$$

Simetrías

En este caso se considera la eliminación de órdenes de escenas reversos. Es decir, dado un orden de escenas, filmar esas escenas al revés tendrá el mismo costo. En la documentación de minizinc, sección 2.6, prácticas de modelamiento efectivo en minizinc, se encuentra una restricción que permite eliminar las soluciones que son un ordenamiento distinto de una solución. En este caso, el reverso, mediante:

$$lex\_lesseq(orden\_escenas, reverse(orden\_escenas))$$

Con la búsqueda predeterminada de minizinc, pasa de 4260 a 340 nodos. Con búsqueda sobre la variable orden_escenas, estrategia de selección de variable first_fail, y selección de valor indomain_min, pasa de 36269 a 17687 nodos.

Heurísticas de Búsqueda

Estas pruebas se realizaron con distintos archivos, pero los resultados descritos a continuación usarán el archivo Med2-5.

Se realiza comparativa de búsqueda usando distintas variables, la estrategia de selección de variable first_fail, y la estrategia de selección de valor indomain_min. En este caso, la variable para la cual el árbol de búsqueda tiene el menor número de nodos por mucho es start_time, que en comparación con la segunda mejor, orden_escenas, tiene aproximadamente el 10% de los nodos.

Ahora, para esta variable, y la estrategia de selección de valor indomain_min, se obtuvo los siguientes resultados para distintas estrategias de selección de variable: para input_order se obtuvo un árbol de búsqueda con 24833 nodos; para first_fail, 1602; para smallest, 1232; y para dom_w_deg, 1014.

Ahora, probando distintas estrategias de selección de valor, tomando como estrategia de selección de variable dom_w_deg, se obtuvo los siguientes resultados: para la estrategia de selección de valor indomain_min se obtuvo un árbol de búsqueda con 1014; para indomain_median, 2992; para indomain_random, 2021; y para indomain_split, 1086.

Finalmente, se realiza la búsqueda sin especificar una estrategia de búsqueda, dejando que el solver decida sobre cuál variable debería buscar, cómo seleccionarla, y cómo seleccionar su valor, y se obtiene un árbol de búsqueda con 340 nodos.

Aplicación

Esta aplicación se desarrolló usando tecnologías web, **Angular** para el frontend, y **Node.js**, específicamente el framework **express** para el backend.

En resumen, esta aplicación ofrece una interfaz gráfica que permite introducir los datos del modelo de forma más cómoda para personas que no conocen el formato que deben tener los archivos para ser reconocidos por minizinc, y una alternativa para quienes prefieren introducir estos datos como texto.

La aplicación construye el texto con los datos y los envía al servidor. El servidor crea un subproceso de minizinc que busca una solución, y cuando encuentra una solución, la regresa al cliente, que la usa para construir la gráfica. Este subproceso de minizinc tiene un tiempo límite de cinco minutos.

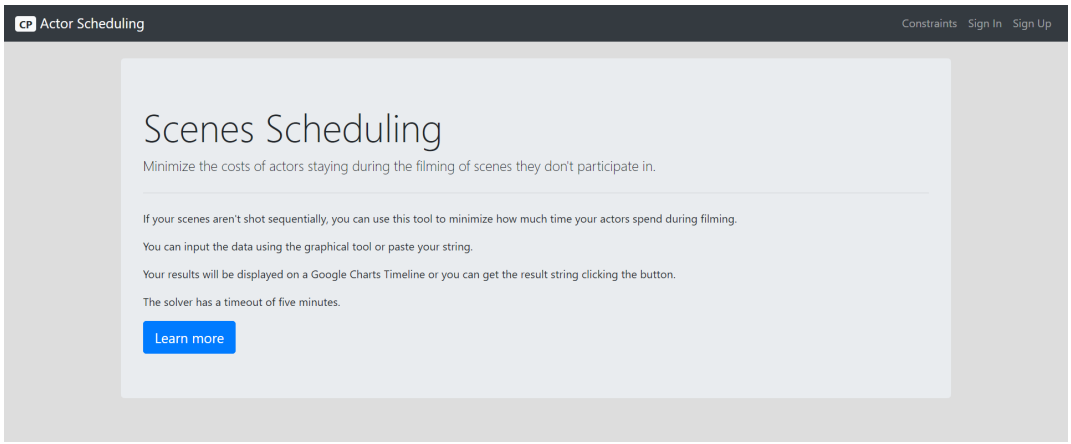

Figure 2: Homepage

La figura 2 muesta cómo se ve la página de inicio del aplicación, la cual contiene una breve descripción de la misma.

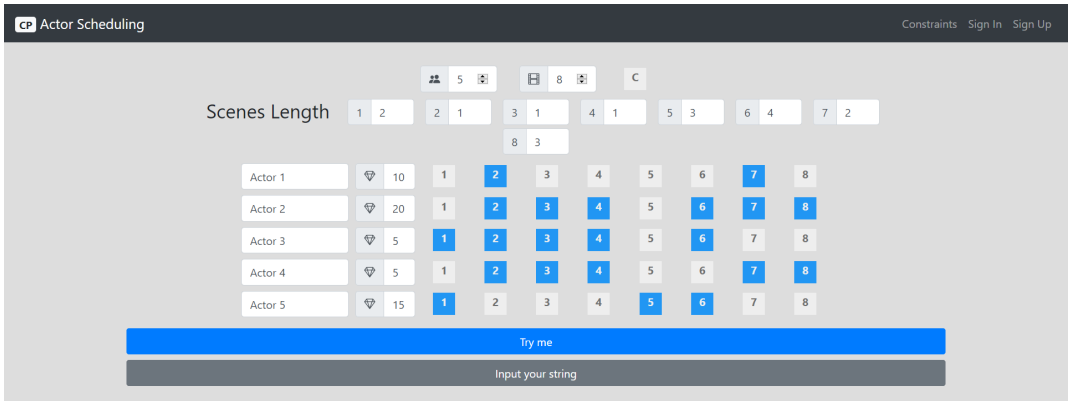

Figure 3: Herramienta gráfica para el modelo número uno

La figura 3 muestra la entrada de uno de los archivos de datos provistos con el planteamiento del proyecto para realizar pruebas. Estas entradas serán convertidas en un string que pueda ser interpretada por minizinc, como se muestra en la figura 4.

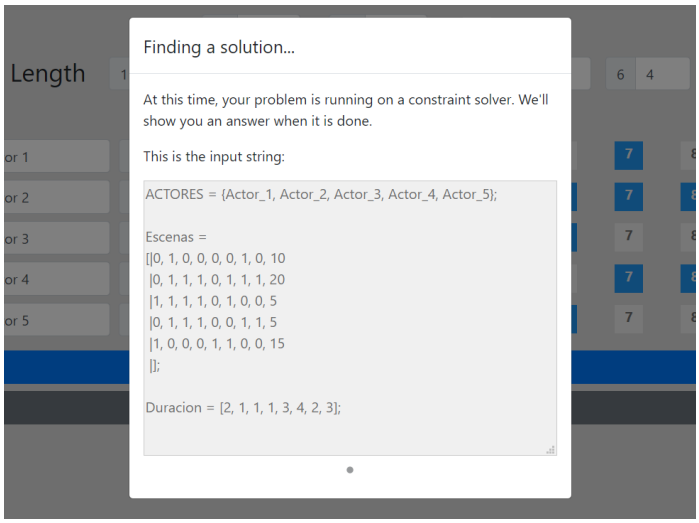

Figure 4: Texto generado por la aplicación.

El texto construido por la aplicación es prácticamente igual a los datos provistos con el planteamiento del proyecto, a excepción de algunos caracteres no imprimibles, como se muestra en la figura 5.

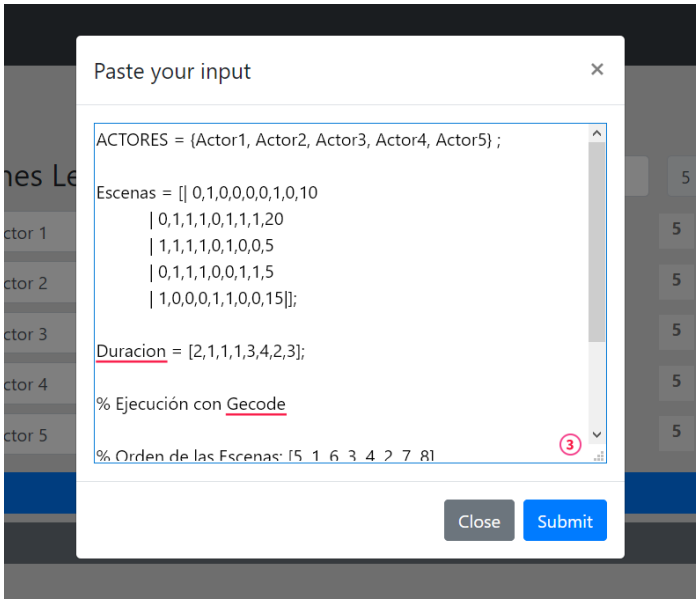

Figure 5: Entrada manual para el modelo uno.

En la figura 6 se puede ver la solución que muestra la aplicación. En la figura 7 se puede ver la solución en forma de texto.

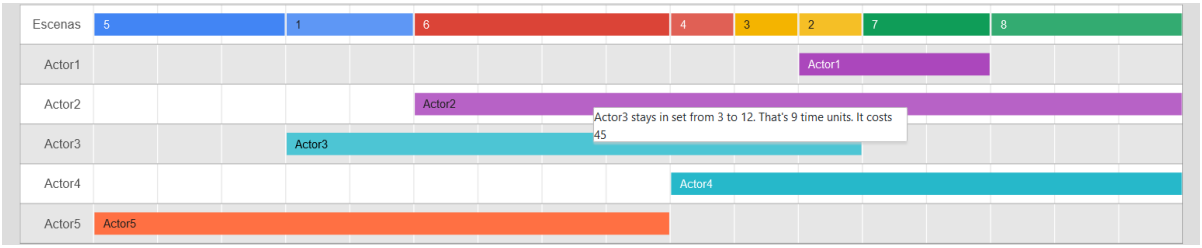

Figure 6: Solución presentada en línea de tiempo para el primer ejemplo.

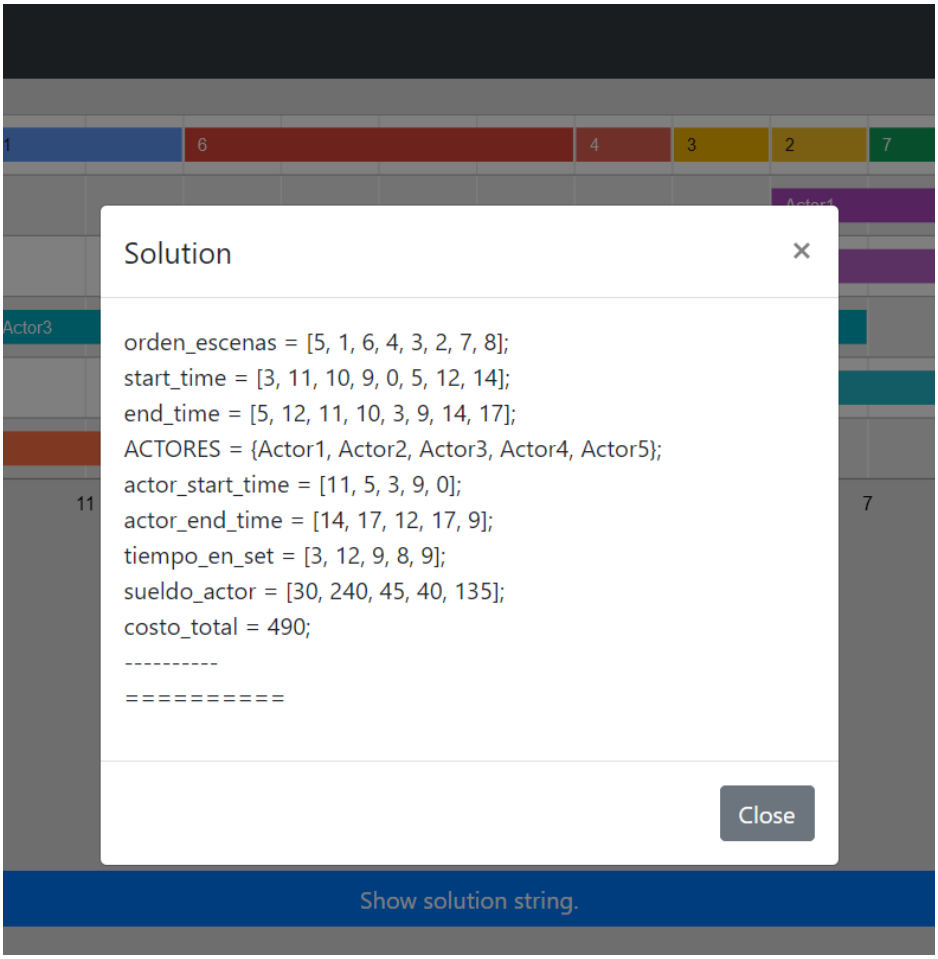

Figure 7: Solución presentada en forma de texto para el primer ejemplo.

Todas estas funcionalidades también están disponibles para el modelo número dos. Una particularidad de la aplicación es que para la entrada de texto, la aplicación busca que la palabra Evitar esté presente en la entrada de texto para determinar si usar el modelo uno o el modelo dos.

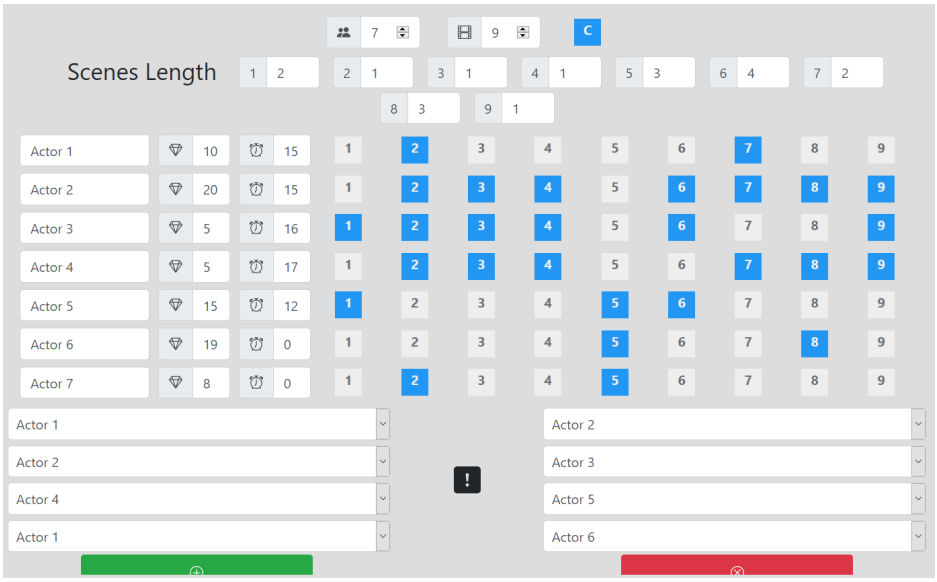

Figure 8: Herramienta gráfica para el modelo número dos

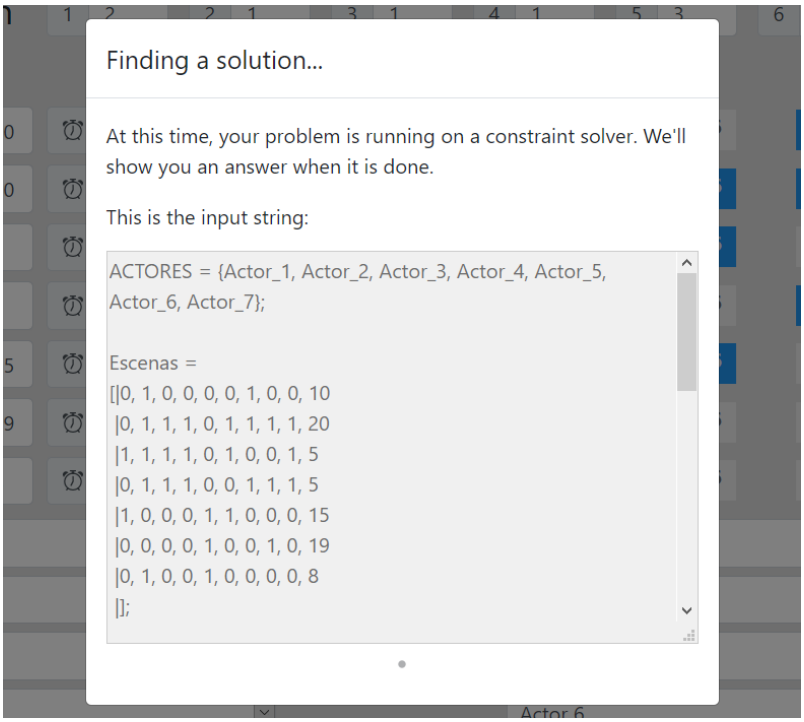

Figure 9: Texto generado por la aplicación.

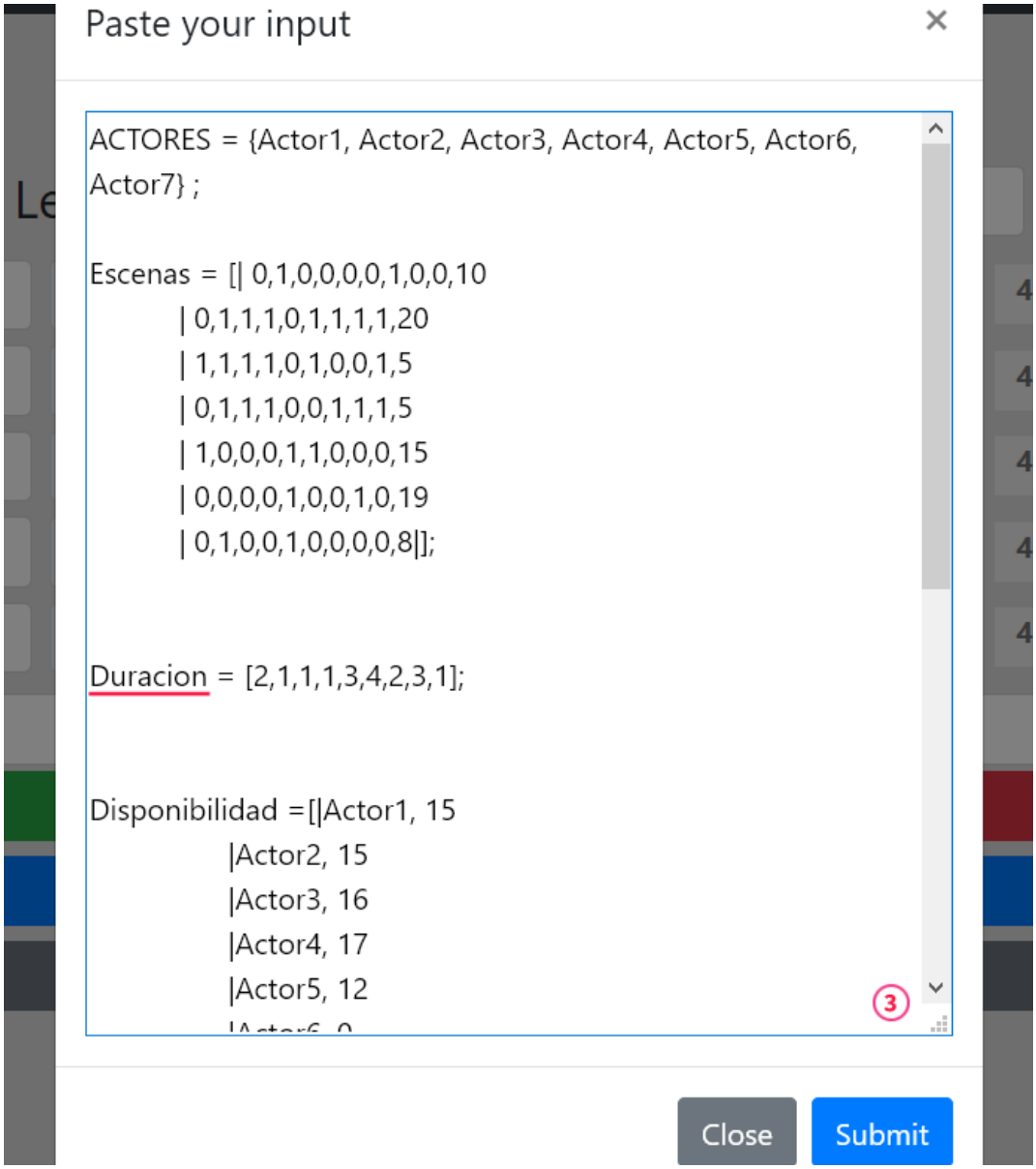

Figure 10: Entrada manual para el modelo dos.

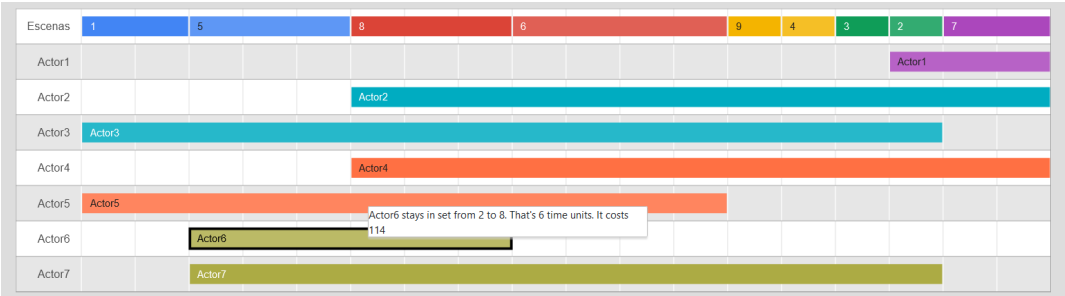

Figure 11: Solución presentada en línea de tiempo para el segundo ejemplo.

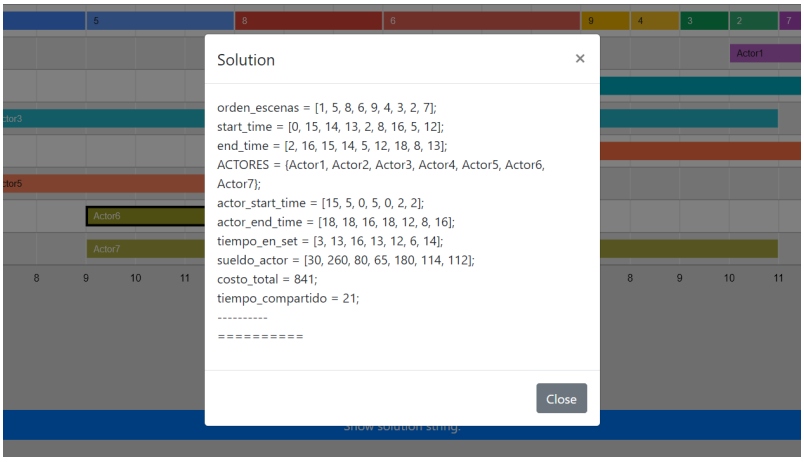

Figure 12: Solución presentada en forma de texto para el segundo ejemplo.