TB1: Tópicos en Ciencias de la Computación

Problema de la Orquesta



Integrantes:

Natalia Maury  
Joaquín Galván  
Carlos Iparraguirre

Profesor

Willy Ugarte

Ciclo: 2022 - 01

**Introducción:**

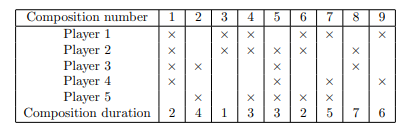
Constraint Programming es encontrar la solución al problema dado cumpliendo todas sus restricciones. Para ello, se utiliza un modelo que contiene las variables, restricciones, dominio de las variables, etc. Esta es una fase muy importante ya que define el problema con sus características que permitirán encontrar una solución. Una vez obtenido el modelo, se utiliza un solver para que lo resuelva, el resultado es la(s) solución(es) al problema. El solver utilizará diversas técnicas como IA, backtracking, etc. para resolver el modelo, y al existir múltiples solvers en diferentes lenguajes de programación, no se creará un software de cero sino se utilizarán los existentes.

Las características de constraint programming es que es procedural y declarativo al mismo tiempo. Al ser declarativo permite indicar los constraints (restricciones) de la solución y al ser procedural permite escribir un programa computacional. Utiliza global constraints, restricción que afecta a un conjunto de variables, para reducir líneas de código y complejidad algorítmica. Se utiliza filtering para reducir el dominio de variables y propagation que consiste en pasar el dominio actualizado al siguiente constraint en la siguiente iteración.

Existen dos tipos de constraint programming, CSP y COP. Constraint Satisfaction Problem (CSP) es un problema donde se deben encontrar todas las soluciones al problema y que todas las restricciones sean cumplidas. Constraint Optimization Problem (COP) es un problema donde se busca la solución más óptima que cumpla con todas las restricciones.

El objetivo del trabajo es aplicar los conceptos de constraint programming con el fin de resolver el problema del orden de la orquesta. El problema de la orquesta consiste en que hay 9 composiciones musicales en las cuales participan ciertos integrantes del equipo. Cada integrante puede llegar inmediatamente antes de que inicie su primera parte e irse inmediatamente después de que termine su última parte.

El objetivo del problema es hallar el orden óptimo de las composiciones musicales en el cuál se minimice el tiempo de espera de los 5 integrantes de la orquesta. Se nos da una tabla con los participantes en cada composición y su duración respectiva. Una X representa si tal integrante participa en tal composición.



Solo hay dos restricciones en este problema: La composición 2 debe tocarse antes de la 8 y la composición 6 debe tocarse inmediatamente después de la 5. El tiempo de espera se considera en cada composición que un integrante no participa y esté presente en el lugar. Por ejemplo, si las composiciones se dieran en el orden del recuadro, el primer integrante esperaría por 14 horas en total (4 + 3 + 7). El quinto esperaría por 1 hora, ya que no debe llegar a la primera composición y puede irse inmediatamente después de la séptima. Este problema, no requiere mostrar todas las combinaciones posibles por lo que sólo se buscará una solución. Además, no tiene una solución óptima por lo que no se aplicará COP ni CSP, sólo constraint programming.

**Problema**

Para poder resolver el problema, se debe tener claro cuál es su objetivo para identificar las variables y su dominio, y tener la representación adecuada. En este caso, las variables son el número de composiciones y el dominio es el orden que puede tener cada composición. Cómo son nueve composiciones, el dominio será de [1,9].

Para representar al número de composiciones y saber qué jugadores participan en ella, se utilizará una matriz booleana donde 1 es que el jugador participa en la composición y 0 es que el jugador no participa en la composición. Debido a que son 5 jugadores y 9 composiciones, las dimensiones de la matriz son 5x9 (cada fila representa a un jugador y cada columna representa una composición).

La matriz booleana de las composiciones tendrá los siguientes valores:

| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |

Para reducir la complejidad algorítmica, los valores de la matriz se almacenan en un csv que leerá el notebook y así sea más rápido el proceso.

Para representar el tiempo de duración de cada composición, se utilizará un vector de enteros. En este vector, cada posición será una composición por lo que su tamaño será 9 y el valor será la duración de dicha composición.

El vector quedaría así en el programa y se añade manualmente en el notebook:

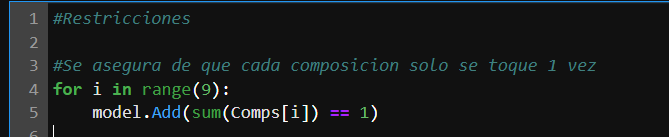
| 2 | 4 | 1 | 3 | 3 | 2 | 5 | 7 | 6 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |

Para la construcción del modelo, se deben primero agregar ciertas variables que serán propias de éste. Primero, se requiere una matriz que contiene el orden y las composiciones, esta matriz consistirá de 9 filas y 9 columnas. Cada fila representa una composición y cada columna representa el orden.

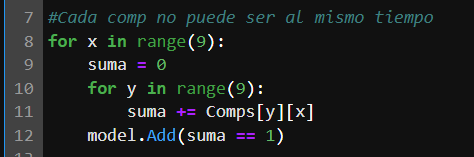


Los pasos que seguirá el modelo para encontrar la solución son los siguientes. Primero, se debe crear el modelo. Luego, agregar y crear todas las variables en el modelo. Luego, agregar todas las restricciones para poder encontrar la solución.

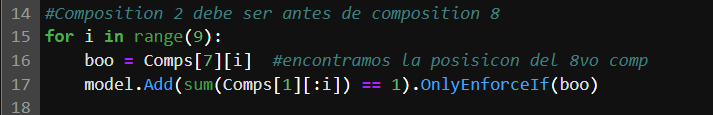
Primero tenemos la restricción de que cada composición solo se puede tocar una vez.



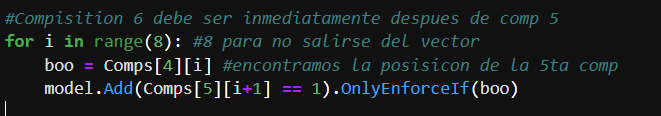
Segundo tenemos que solo se puede tocar una composicion al mismo tiempo.



Tercero tenemos que la composicion 2 debe ir antes de la composicion 8.

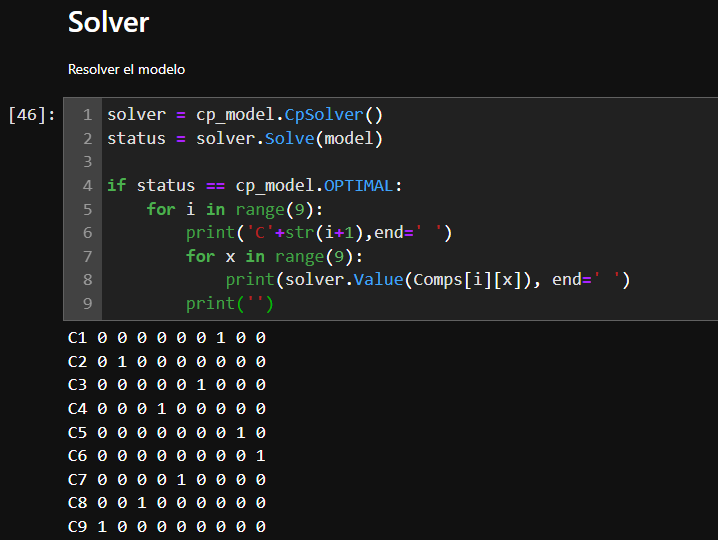


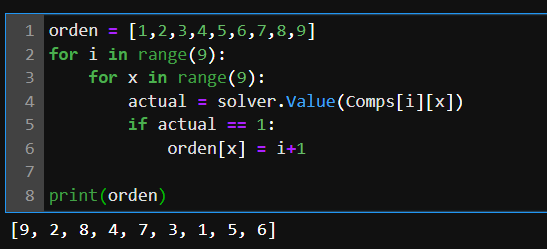
Cuarto tenemos que la composición 6 debe ser tocada inmediatamente después de la composición 5.



Para obtener el tiempo mínimo, primero se debe pasar el orden obtenido de la matriz pasarlo a un vector para poder realizar los cálculos de los tiempos de manera más rápida. Esto se haría considerando que la columna tiene el orden por lo que, se buscará cuál matriz[i][j] sea 1, si es así, entonces el vector en la posición j almacenará la composición i. De esta manera, se pueden obtener las composiciones y su orden con la matriz. Luego, se consultará la matriz booleana de composiciones para obtener los tiempos de espera de cada participante. El tiempo de espera de cada participante será almacenado en un vector. La idea para calcular el tiempo de espera era que si se hallaba un 0 entre dos unos, se sumaba el tiempo de espera. Si no hay un número uno a cada lado, no se considera el tiempo de espera. Se sumarán los tiempos de cada participante en una variable, y ese será el tiempo total de espera.

**Resultados:**





El modelo encuentra el orden para cada composición, pero faltó el cálculo del tiempo de espera.

Link del Código: <https://drive.google.com/file/d/12p0nNNVHNuT1MXy3mwR_HL_70xe4Kuqi/view?usp=sharing>

Link de la presentación:

<https://docs.google.com/presentation/d/1Fle-PQhJSYljTwz_kYjtMmU76f_L1YOTVcmwt889hac/edit#slide=id.p>

**Conclusiones:**

El modelo funcionó bien hasta donde se logró desarrollar, con un poco más de tiempo pudo llegar a completarse y resolver correctamente el problema de la orquesta calculando los tiempos.