

**Universidad Peruana de Ciencias Aplicadas (UPC)**

**Ciclo 2020-1**

**CC58 - Tópicos en Ciencias de la Computación**

**Sección: CC71**

**TRABAJO PARCIAL**

**Tema:**

**Modelo de Programación con restricciones para el problema Orchestra rehearsal**

**PROFESOR DEL CURSO:** Willy Gustavo Ugarte Rojas

**TRABAJO PRESENTADO POR LOS ALUMNOS:**

1. Castillo Arredondo, Geral Esteen

2. Zegarra Garcia, Andre Prince Francois

Monterrico, 7 de mayo del 2020

**Índice:**

1. **Definición del problema**

* **Conceptualización del problema**
* **Constantes del problema**

1. **Descripción del modelo**
2. **Variables y Dominios**

* **Variables simples**
* **Tablas y arreglos de variables**

1. **Restricciones**

* **Restricciones sin reification**
* **Restricciones con reification**
* **Eliminación de simetría**

1. **Heurísticas**

* **Uso de Heurísticas**
* **Comparación entre distintas Heurísticas**
* **Explicación de los mejores resultados**

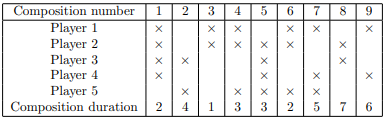
1. **Función de optimización**
2. **Conclusiones**
3. **Definición del problema**

* **Conceptualización del problema**

Este problema aborda lo relacionado a la programación de un concierto a nivel de el orden respectivo de cada composición. Se parte del hecho de que un concierto está compuesto por un número determinado de composiciones. Conforme a ello, el concierto consta de 9 composiciones musicales. Asimismo, cada composición implica 5 miembros de la orquesta. A continuación, se listan las condiciones y características relacionadas al problema:

* Cada músico puede llegar inmediatamente antes del primera composición en la que participa y se va inmediatamente después de su última composición.
* Durante el concierto, la composición 2 debe reproducirse antes de la composición 8.
* Durante el concierto, la composición 6 debe reproducirse inmediatamente después de la composición 5.

Con respecto al valor que se desea **optimizar**, es necesario encontrar el orden de las composiciones para un concierto tal que el tiempo total de espera de los músicos se minimice. Con respecto a los datos del problema, estos serán presentados en una tabla. Se debe considerar que la celda (i, j) de la tabla contiene el valor de x si el jugador i participa en la composición j. En el caso de que no participe, la celda se deja en blanco. La tabla respectiva se muestra a continuación:

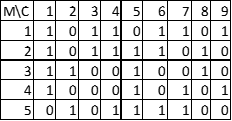


En la última fila de la tabla presentada, se indica la duración de cada composición respectivamente. Llegados a este punto, se brindará un ejemplo para entender mejor cómo se obtiene el tiempo total de espera a optimizar. Si el orden de las composiciones en el concierto fuera (1, 2, 3, 4, 5, 6, 7, 8, 9), el tiempo de espera para:

* Para el jugador 1, es 14 (4 + 3 + 7).
* Para el jugador 2, es 9 (4 + 5).
* Para el jugador 3, es 11 (1 + 3 + 2 + 5).
* Para el jugador 4, es 17 (4 + 1 + 3 + 2 + 7).
* Para el jugador 5, es 1.

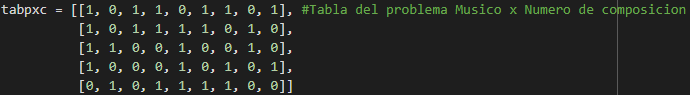
Respecto a ello, El tiempo de espera total es entonces igual a 52. Este valor o parámetro debe ser minimizado. Para la resolución de este problema, se plantea modelar el mismo como un problema de satisfacción de restricciones. Por otro lado, es importante especificar una función de optimización adecuada que permita determinar la mejor solución.

* **Constantes del problema**
* **tabpxc:** Esta tabla no es de variables. Por el contrario es una tabla constante del problema. Asimismo, esta tabla corresponde a aquella encargada de relacionar los músicos con las composiciones. El objetivo principal de esta tabla es indicar en qué composiciones participa el músico y en cuáles no. A continuación, se representa dicha tabla:



En la primera columna, los músicos están identificados con un índice numérico. En la primera fila, se encuentran las composiciones identificadas por su índice numérico respectivo. Dentro de la tabla, se toma el valor de 1 si el músico dado por la fila participa en la composición dada por la columna y 0 si dicho músico no participa en determinada columna.

**Código respectivo:**

****

* **arrcxd:** Este arreglo no es de variables. Se trata de un arreglo que contiene valores constantes dados por el problema. Ahora bien, se procede a representar a dicho arreglo:



En la primera fila, se indican los índices del arreglo correspondientes al índice numérico asignado a cada composición. Ahora bien, en la segunda fila, se representan los valores del arreglo respectivos. Dichos valores son constantes y corresponden al tiempo de duración de cada composición

**Código respectivo:**

****

1. **Descripción del modelo**
2. **Variables y Dominios**

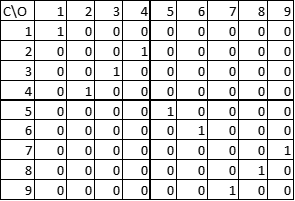
* **Variables simples**
* **waiti:** Esta variable indica el tiempo total de espera de todos los miembros de la orquesta. Dado un orden de composiciones en el concierto. Se debe tener en cuenta que cada músico llega inmediatamente antes de su primera composición. Asimismo, se va inmediatamente después de que termine la última composición en la que este participa. En ese sentido, existen momentos de espera en los que el músico está en el concierto pero no participa en ninguna composición. La duración de cada uno de esos momentos indica el tiempo de espera de cada músico. La suma de todos los tiempos de espera corresponde al valor asignado a esta variable.

**Dominio de variables de la tabla:** [0, 200]

**Código respectivo:**

****

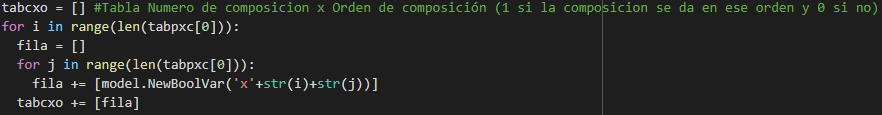
* **Tablas y arreglos de variables**
* **tabcxo:** Se trata de unatabla de variables. Esta tabla indica intersecciones o coincidencias entre las composiciones y su respectivo orden. A continuación, una tabla para explicar lo dicho de manera más clara:



Para este caso, las variables son las que representan los valores de 0 o 1 dentro de la tabla. Los números que se encuentran en la primera columna vertical indican las composiciones, las cuales están representadas por un número. En este caso, las composiciones no tienen nombre, pero se identifican por un índice numérico. Ahora bien, la primera fila superior indica el orden en el que cada composición tomará lugar respectivamente. En la tabla, es posible notar que las intersecciones entre índice de composición y orden de composición pueden representarse con 1 o 0. El 1 indica correspondencia entre índice y orden, mientras que el 0 indica lo contrario. En el caso presentado, por ejemplo, se tendría el siguiente orden: La composición 1 sería primera, la composición 4 sería segunda, la 3 sería tercera, la 2 sería cuarta y así correspondientemente.

**Dominio de variables de la tabla:** [0, 1]

**Código respectivo:**



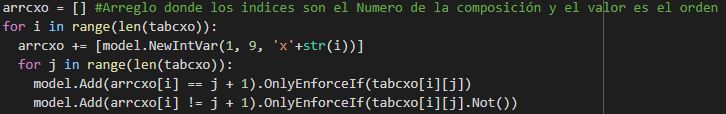
* **arrcxo:** Es un arreglo de variables. Este arreglo indica el orden de cada composición. Asimismo, está estrechamente relacionado con la tabla **tabcxo**. Se procede a presentar un arreglo para mejor entendimiento.



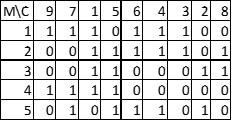
La primera fila representa los índices del arreglo y los índices de cada composición. Además, la segunda fila representa los valores del arreglo y el orden que le corresponde a cada composición respectivamente. En ese sentido, es factible identificar que esta estructura indica lo mismo que la estructura anterior. No obstante, su existencia se justifica, ya que permite realizar un mejor manejo, control y ejecución de ciertas restricciones respeto al orden de las composiciones.

**Dominio de variables del arreglo:** [1, 9]

**Código respectivo:**

****

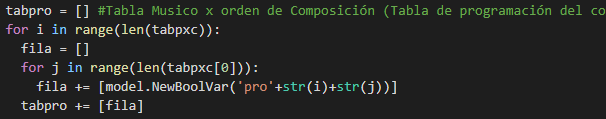
* **tabpro:** Se trata de una tablaque representa la tabla de programación del concierto. Las variables permiten identificar la relación entre músico y composición, estando las composiciones respectivamente ordenadas. Ello se representa como sigue:



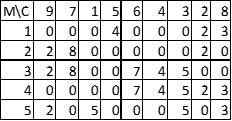
En la primera columna, se encuentran identificados los músicos con un índice numérico. En la primera fila, las composiciones están ordenadas con respecto al orden indicado en las estructuras **arrcxo** y **tabcxo**. Asimismo, están representadas por su indice numerico para identificarlas mejor. Las variables son aquellas encargadas de indicar las relación entre músico y composición con el valor de 0 o 1. En este caso, dicho valor es 1 si el músico participa de la composición respectiva y 0 si no.

**Dominio de variables de la tabla:** [0, 1]

**Código respectivo:**

****

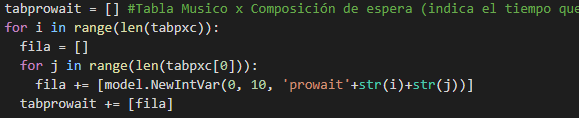
* **tabprowait:** Corresponde a una tabla de variables que representa la programación del concierto. No obstante, en este caso, la tabla indica el tiempo que debe esperar el músico en una composición si no participa en ella. A continuación, una ejemplificación:



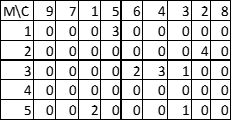
Esta tabla es parecida a la anteriormente presentada, sin embargo se debe especificar las diferencias respecto a los valores que toman las variables dependiendo del caso. Si el músico participa en la composición, entonces la variable respectiva toma el valor de 0. Si el músico no participa en la composición, la variable toma el valor del tiempo de espera del músico si este se queda en el concierto. En otras palabras, dicho valor corresponde al tiempo de duración de la composición respectiva.

**Dominio de variables de la tabla:** [0, 10]

**Código respectivo:**

****

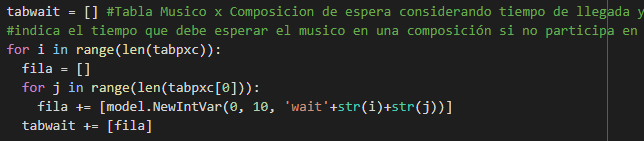
* **tabwait:** Se trata de una tabla de variables. Esta tabla tiene como objetivo indicar el tiempo que debe esperar el músico en una composición si no participa en ella considerando tiempo de llegada y salida. Ahora bien, el ejemplo respectivo:



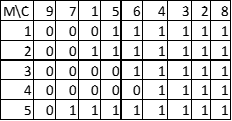
Esta tabla parte de una restricción dada en la definición de problema, en la cual se argumenta que el miembro de la orquesta o músico llega justo a tiempo para la primera composición en la que participa y se va apenas termine la última. En ese sentido, las variables de la tabla toman el valor de 0 si el músico ya se fue del concierto, si todavía no llega o si participa de la composición. Por otro lado, si el músico está presente durante dicha composición, entonces la variable toma como valor el tiempo de duración de la composición correspondiente.

**Dominio de variables de la tabla:** [0, 10]

**Código respectivo:**

****

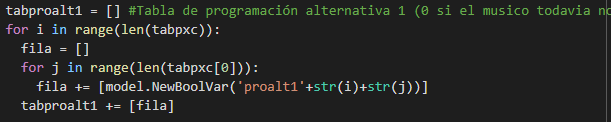
* **tabproalt1:** Es la primera tabla de programación con variables alternativa. El ejemplo es el siguiente:



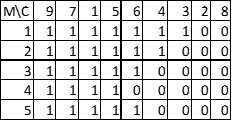
De acuerdo a esta tabla, las variables toman el valor de 0 si el músico todavía no llega cuando se conduce cierta composición. Ahora bien, si el músico ya había llegado, se toma el valor de 1. Esta tabla sirve como soporte en la construcción de restricciones que permitan determinar la tabla **tabwait**.

**Dominio de variables de la tabla:** [0, 1]

**Código respectivo:**

****

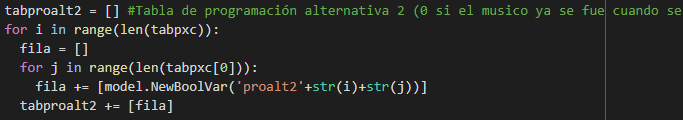
* **tabproalt2:** Es la segunda tabla de programación con variables alternativa. El ejemplo es el que sigue:



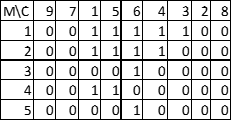
De acuerdo a esta tabla, las variables toman el valor de 0 si el músico ya se fue cuando se conduce la composición respectiva y 1 si todavía no se va o no llega en dicho momento. Esta tabla es útil en la construcción de restricciones que permitan darle valor a las variables de la tabla **tabwait**.

**Dominio de variables de la tabla:** [0, 1]

**Código respectivo:**

****

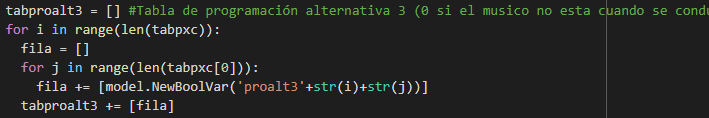
* **tabproalt3:** Es la tercera tabla de programación con variables alternativa. Se representa así:



Con respecto a esta tabla, las variables toman el valor de 0 si el músico no está cuando se conduce la composición respectiva y 1 si sí está dicho músico. Esta tabla sirve como soporte para el desarrollo de restricciones, las cuales tienen como objetivo determinar los valores de la tabla **tabwait**.

**Dominio de variables de la tabla:** [0, 1]

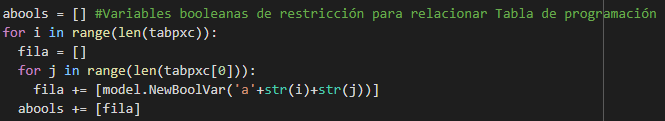
**Código respectivo:**

****

* **abools:** Se trata de una tabla o matrix de variables de tamaño 5x9. Esta matrix contiene variables booleanas usadas en la construcción de “reified constraints”. Este tipo de restricciones sirven para relacionar Tabla de programación (**tabpro**) con Tabla alternativa 1 (**tabproalt1**).

**Dominio de variables de la tabla:** [0, 1]

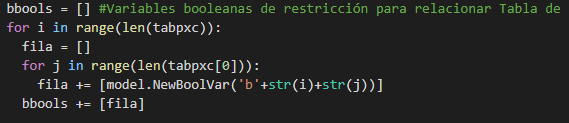
**Código respectivo:**

****

* **bbools:** Corresponde a una tabla o matrix de variables de tamaño 5x9. Asimismo, se compone de variables booleanas usadas en la construcción de “reified constraints”. Dichas restricciones se encargan de relacionar Tabla de programación (**tabpro**) con Tabla alternativa 2 (**tabproalt2**).

**Dominio de variables de la tabla:** [0, 1]

**Código respectivo:**



1. **Restricciones**

* **Restricciones sin reification**

A continuación, se procederá a listar las restricciones con su respectivo nombre y a explicar cada una de ellas:

* **Cada composición debe tener un orden distinto:** Esta restricción permite establecer que el orden asignado a cada composición en el concierto debe ser distinto. Esta restricción se aplica sobre el arreglo de variables **arrcxo**, indicando que todos sus valores deben ser diferentes.

**Código respectivo:**

****

* **La composición 2 debe estar antes que la composición 8:** Con esta restricción, se busca fijar la condición de que el orden asignado a la composición 2 debe ser inferior al orden asignado a la composición 8. De esta manera, la composición 2 va a tocarse primero en el concierto. Para ello, se debe acceder a los valores de **arrcxo** por medio de índices respectivos y aplicar la condición descrita. Los índices utilizados son 1 y 7, ya que en los arreglos el primer indice es 0 y no 1.

**Código respectivo:**



* **La composición 6 debe darse justo después de la 5:** Esta restricción busca establecer que el orden de la composición 5 debe ser igual al orden asignado a la composición 6 incrementado en una unidad. Es decir, si la composición 5 es primera, la 6 debe ser segunda. Esto se logra accediendo a los valores de **arrcxo** con los indices respectivos y estableciendo la condición. Los índices utilizados son 4 y 5, ya que en los arreglos el primer indice es 0 y no 1.

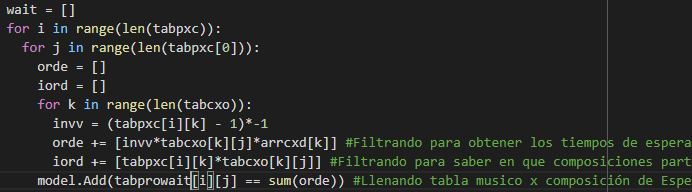
**Código respectivo:**

****

* **Llenado de la tabla musico x composición de espera:** Estas restricciones son utilizadas como una operación que permita determinar los valores de la tabla **tabprowait**. Para ello, se sigue un proceso, el cual consiste en los siguientes pasos:

1. Invertir los valores de la matriz o tabla **tabpxc**. Es decir, si su valor fue 1, entonces ahora será 0 y viceversa.
2. Cada fila de **tabpxc** invertida debe multiplicarse por el arreglo **arrcxd**. Esta operación se realiza multiplicando el primer valor de la fila respectiva de la tabla **tabpxc** por el primer valor del arreglo mencionado, y así sucesivamente. De esta manera, se filtran solo las duraciones de composiciones correspondientes a esa fila. En otras palabras, sólo aquellas composiciones en las que el músico, respectivo a esa fila, no participará y tendrá que esperar.
3. Luego de ello, los valores de cada fila resultado de la operación anterior deben multiplicarse por los valores de cada columna de la tabla **tabcxo**. Esta operación se realiza multiplicando el primer valor de la fila respectiva con el primer valor de la columna, y así sucesivamente.
4. Cuando se multiplique determinada fila con determinada columna, el resultado de dicha operación será un arreglo. Se debe sumar los valores de ese arreglo. El resultado de esa suma será ubicado en la tabla **tabprowait**. La fila y columna de dicha tabla que le correspondan a este resultado debe coincidir con el índice de la fila y la columna de la operación respectiva. Con esto, se hace referencia a las filas y columnas que fueron multiplicados para llegar a ese valor.
5. De esta manera, se completa o determina la tabla **tabprowait**. Las dimensiones de esta tabla son 5x9, ya que **tabpxc** presenta 5 filas y **tabcxo** presenta 9 columnas.

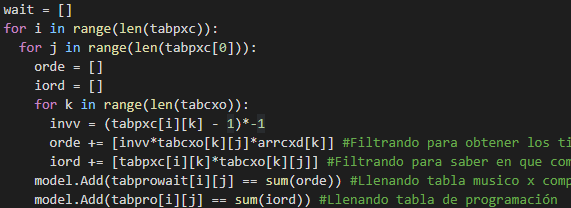
**Código respectivo:**



* **Llenado de tabla de programación:** Al igual que en el caso de las restricciones anteriores, se busca determinar los valores de la tabla **tabpro**. En ese sentido, se sigue el siguiente procedimiento:

1. Se multiplican los valores de cada fila de la tabla **tabpxc** por los valores de cada columna de la tabla **tabcxo**. Esto último se realiza, de tal manera que el primer valor de determinada fila de **tabpxc** se multiplique por primer valor de determinada columna de **tabcxo**, y así sucesivamente.
2. La multiplicación de cierta fila con cierta columna da como resultado un arreglo. Se deben sumar los valores de dicho arreglo. Dicha suma será puesta en la tabla **tabpro**. La posición de este valor en la tabla mencionada está dada por los índices correspondientes a la fila y columna de las tablas multiplicadas para llegar a dicho valor.
3. De esta manera, se determina la tabla **tabpro**. Dicha tabla cuenta con dimensiones 5x9, puesto que **tabpxc** está dada por 5 filas y **tabcxo** presenta 9 columnas.

**Código respectivo:**



* **Llenado de tabla de programación alternativa 3:** Las restricciones para este caso poseen como objetivo determinar los valores de la tabla **tabproalt3**. Para ello, se multiplica cada valor de la tabla **tabproalt1** con cada valor de la tabla **tabproalt2**, y el resultado es ubicado en la tabla **tabproalt3** respectivamente. Esto se realiza siguiendo un orden de índices. En ese sentido, por ejemplo, el valor de la fila 1 y columna 1 de la tabla **tabproalt1** se multiplica por el valor de la fila 1 y columna 1 de la tabla **tabproalt2**, y el resultado es ubicado en la fila 1 y columna 1 de la tabla **tabproalt3**. Este procedimiento se realiza exactamente igual para determinar el resto de valores de la tabla **tabproalt3**. Cabe mencionar que el código a mostrar se encuentra dentro de un doble bucle, en el que van cambiando los índices i y j.

**Código respectivo:**



* **Llenado de Tabla de Espera considerando tiempo de llegada y salida:** Para este caso, las restricciones poseen como objetivo el cálculo de los valores de la tabla **tabwait**. Para ello, se multiplica la tabla **tabproalt3** con la tabla **tabprowait**. El resultado es ubicado en la tabla **tabwait**, y se obtiene con el mismo procedimiento explicado en las restricciones del caso anterior. Asimismo, el código a mostrar se encuentra dentro de un doble bucle, en el que se van modificando los índices i y j.

**Código respectivo:**



* **Determinado de tiempo total de espera:** Con esta restricción se pretende establecer que la variable **waiti** debe tener el valor correspondiente al tiempo total de espera de todos los músicos o miembros de la orquesta. Para ello, se suman los valores de cada fila de la tabla **tabwait** y cada suma se almacena en el arreglo **wait**. La suma de los valores del arreglo **wait** corresponde a el tiempo total de espera. Dicho suma es asignada a la variable **waiti**.

**Código respectivo:**

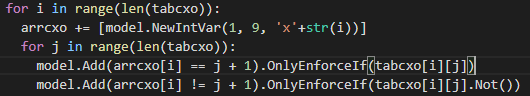


* **Restricciones con reification**
* **Relación entre arrcxo y tabcxo:** Estas restricciones lo que buscan es sincronizar las tablas **tabcxo** y **arrcxo**. Ambas tablas relacionan las composición con su respectivo orden en el concierto. En ese sentido, ambas tablas deberían indicar los mismo. En el arreglo **arrcxo**, los índices identifican las composiciones y los valores del arreglo indican el orden de cada composición respectivamente. En la tabla **tabcxo**, las filas son las composiciones y las columnas el orden respectivo. Por ello, en la matriz booleana se indica 1 si la composición (fila) se da en el orden (columna) y 0 si no. De acuerdo a las descripciones de ambas estructuras, se plantea la siguiente restricción:

***(valor de arrcxo en a posición i =j+1) ⇔ (valor de tabcxo en las posiciones i y j =1***)

En la primera condición, se utiliza j+1, ya que los índices empiezan en 0 en las matrices y arreglos, y el dominio de **arrcxo** es [1, 9].

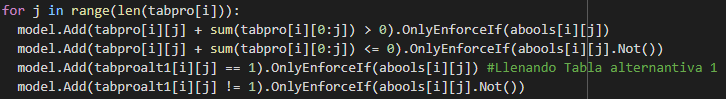
**Código respectivo:**



* **Llenado de tabla de programación alternativa 1:** Las restricciones del caso pretenden determinar los valores de la tabla **tabproalt1**. Se debe considerar que dicha tabla booleana relaciona a los músicos con las composiciones, indicando 0 si el músico todavía no llega cuando se conduce cierta composición y 1 si ya había llegado. Por lo tanto, es factible plantear la siguiente restricción, dada la naturaleza de esta estructura:

***(valor de tabproalt1 en las posiciones i y j = 1) ⇔ (suma de los valores de tabpro en la fila i y hasta la posición j > 0)***

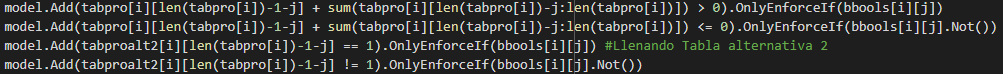
**Código respectivo:**

****

* **Llenado de tabla de programación alternativa 2:** En este caso, las restricciones pretenden determinar los valores de la tabla **tabproalt2**. Se debe considerar que dicha tabla booleana relaciona a los músicos con las composiciones, indicando 0 si el músico ya se fue cuando se conduce cierta composición y 1 si todavia no se va o no llega. En ese sentido, es posible plantear la siguiente restricción, dada la función de esta estructura:

***(valor de tabproalt2 en las posiciones i y (8-j) = 1) ⇔ (suma de los valores de tabpro en la fila i y desde la posición (8-j) hasta la posición 8 > 0)***

**Código respectivo:**

******

* **Eliminación de simetría**
* **Eliminando simetría en arrcxo:** El objetivo de esta restricción se basa en eliminar soluciones simétricas. En otras palabras, se trata de eliminar soluciones que representen lo mismo que las ya encontradas pero en diferente orden. Esta técnica reduce el espacio de búsqueda y la complejidad algorítmica del modelo. Por ello, se evita la simetría en la distribución de valores en el arreglo **arrcxo**. A continuación, se plantea un ejemplo para entender mejor esta condición. Una solución del arreglo **arrcxo** podría ser así:



Asimismo, una solución simétrica a la anterior sería así:



Se busca que el modelo evite evaluar soluciones simétricas o equivalentes como las mostradas anteriormente. Para ello, se plantea como restricción que el primer valor del arreglo **arrcxo** correspondiente al índice 0 debe ser menor o igual a 4. Ello se representa así:

***(valor del arreglo arrcxo en la posición 0) <= 4***

**Código respectivo:**



1. **Heurísticas**

* **Uso de Heurísticas**

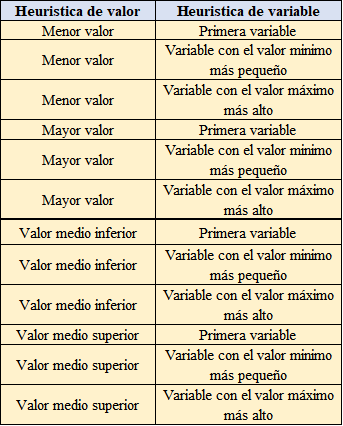
Con respecto a las heurísticas utilizadas, es apropiado dividir las mismas en heurísticas de variable y heurísticas de valor. Con respecto a heurísticas de variable, se aplicaron las siguientes:

* **Primera variable:** En este caso, como su nombre lo indica, la heurística se basa en escoger la primera variable que se encuentre o que se instancia dentro del modelo.
* **Variable con el valor mínimo más pequeño:** Con respecto a esta heurística, entre las variables no vinculadas, se debe seleccionar la variable con el valor mínimo más pequeño. En caso de empate, se selecciona el primero, "primero" definido por el orden en el vector de **IntVars** utilizado para crear el selector.
* **Variable con el valor máximo más alto:** La heurística se basa en, entre las variables no vinculadas, seleccionar la variable con el valor máximo más alto. Si se presenta empate, se debe seleccionar el primero, primero definido por el orden en el vector de **IntVars** utilizado para crear el selector.

Ahora bien, es apropiado proceder a definir y conceptualizar las heurísticas de valor aplicadas en este caso. Con respecto a ello, se aplicaron las siguientes:

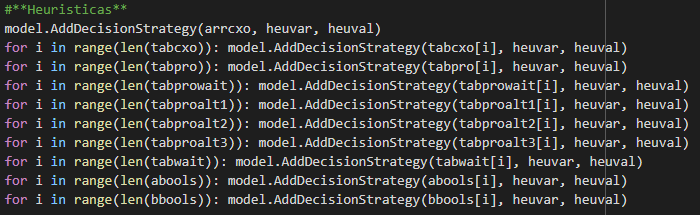
* **Menor valor:** Cuando se le asigna los valores a una variable, se debe priorizar asignar los valores más pequeños o menores posibles dentro del dominio respectivo.
* **Mayor valor:** De forma análoga a como se aplica para la heurística anterior, en este caso, se debe priorizar asignar los valores más altos o mayores posibles. Dichos valores deben encontrarse dentro del dominio respectivo de la variable.
* **Valor medio inferior:** Con respecto a la asignación de valores por medio de esta heurística, en primer lugar, se debe dividir el dominio de la variable en dos alrededor del centro. Luego de ello, se le asigna a la variable el primer valor de la mitad inferior.
* **Valor medio superior:** Para la asignación de valores, de forma análoga a la heurística anterior, se divide el dominio en dos alrededor del centro. Como último paso, se obliga a la variable a tomar el primer valor en la mitad superior

Continuando con el asunto, se procede a combinar las distintas heurísticas de variable con las distintas heurísticas de valor. De esta forma, se generan 12 combinaciones de heurísticas posibles a comparar. Estas combinaciones serán representadas a continuación:



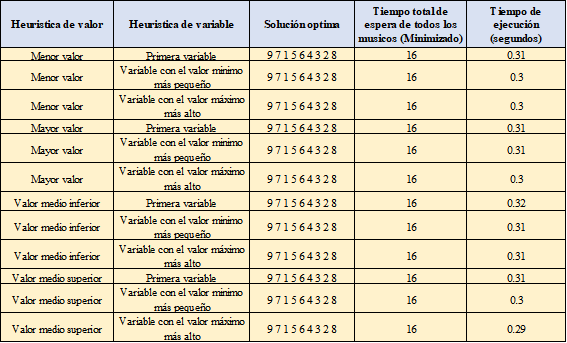
Cabe mencionar que dentro del código de programación, las distintas heurísticas se envían como parámetros y están identificadas con números. Avanzando en el tema, las distintas combinaciones heurísticas planteadas son evaluadas en el modelo. Asimismo, el impacto y los resultados obtenidos por cada heurística serán comparados en el siguiente punto.

**Código respectivo:**



* **Comparación entre distintas Heurísticas**

Se realizó una evaluación de los resultados obtenidos con cada combinación de heurísticas de variable y valor aplicadas. En esta parte, dichos resultados serán comparados con el objetivo de determinar las heurísticas más convenientes. A continuación, se presenta una tabla indicando lo obtenido con respecto a cada estrategia utilizada:



Como se puede ver en la tabla, se muestra la solución óptima, tiempo de espera y tiempo de ejecución para cada combinación de heurísticas evaluada. En cuanto a la solución óptima encontrada, se obtiene la misma solución en cada caso. Asimismo, el tiempo total de espera de todos los músicos óptimo es el mismo para todos los casos analizados. Con respecto al tiempo de ejecución, el valor del mismo no solo depende de la complejidad del modelo, sino también de otras condiciones del sistema de ejecución. En ese sentido, si bien existe una variabilidad con respecto al tiempo de ejecución para cada caso, la misma no es lo suficientemente significativa. Todos lo tiempos obtenidos se aproximan a 0.3, por lo tanto no es posible concluir que existe una diferencia significativa con respecto a la complejidad del modelo aplicando las distintas heurísticas planteadas.

* **Explicación de los mejores resultados**

Los resultados obtenidos con las distintas heurísticas evaluadas son similares a nivel de solución óptima encontrada y tiempo total de espera de todos los músicos. No obstante, con respecto al tiempo de ejecución, el valor obtenido varía con respecto a las heurísticas comparadas. Las condiciones del sistema de ejecución influyen en él variabilidad del tiempo de ejecución de una función. En otras palabras, no solo depende de la complejidad del modelo. En ese sentido, dado que los tiempos de ejecución obtenidos son similares, es posible concluir que el uso de las distintas heurísticas evaluadas no genera un impacto significativo en el tiempo de ejecución. Se ha logrado comprobar que los resultados comparados para todas las heurísticas dadas no presentan mayor diferencia. Por lo tanto, se concluye que no existe ninguna heurística, ya sea de valor o variable, que presente una ventaja representativa con respecto al resto de estrategias analizadas. Dicha ventaja se mide a nivel de solución óptima encontrada y tiempo de ejecución o complejidad del modelo.

1. **Función de optimización**

Con respecto a este problema, la función de optimización se focaliza en minimizar el valor de la variable **waiti**. En este caso, el problema exige que se minimice el tiempo de espera total de los músicos dadas determinadas condiciones. El tiempo de espera total se obtiene sumando el tiempo de espera de cada músico o miembro de la orquesta. Dicho tiempo total se almacena en la variable **waiti**. En ese sentido, la función de optimización se basa en minimizar el valor o estimación que toma dicha variable.

**Código respectivo:**

****

1. **Conclusiones**

En síntesis, se ha logrado la solución del problema definido por medio de un modelo de programación de restricciones con optimización. El problema se basa en determinar la programación de composiciones en un concierto, minimizando el tiempo de espera de los músicos y considerando que estos no se pueden ir del concierto hasta la terminación de la última composición en la que participen. Se plantearon variables, dominios, restricciones y funciones de optimización apropiadas para el problema. La solución fue posible, ya que se usó un solucionador de programación de restricciones. Es importante conocer y aplicar este tipo de paradigmas de programación cuando se analiza un problema, el cual se podría ajustar a las características de un modelo de programación de restricciones.

Asimismo, es importante argumentar que ninguna de las restricciones planteadas puede ser suavizada. Esto es así, ya que cada restricción planteada es indispensable en la solución del problema. La suavización de cualquiera de las restricciones podría representar encontrar soluciones no apropiadas o no óptimas para el problema. Ahora bien, con respecto a cómo se tomaría en cuenta las restricciones suaves para el problema dado, se argumenta que este tipo de restricciones no serían aplicadas de ninguna manera. Es conveniente aplicar restricciones suaves o soft constraints cuando las restricciones con las que se cuentan son demasiado fuertes o se contradicen y no permiten llegar a una solución satisfactoria. En este caso, se ha logrado llegar a una solución satisfactoria y óptima aplicando las restricciones del modelo derivadas del problema dado. En ese sentido, **se concluye que no es posible incluir restricciones suaves o suavizadas dada la naturaleza del problema**.