Proyecto: El Problema de la Planificación del Calendario de Torneos Deportivos (CalDep)

Laura Daniela Jaimes - 2040430

Santiago Casañas Tabares - 2024301

Mayra Alejandra Sanchez - 2040506

Jesús Adrian Peña - 2025513

Juan Francisco Diaz
Jesús Aranda

Análisis y diseño de algoritmos 2

El problema

El problema a resolver en este proyecto es la creación de calendarios adecuados para torneos deportivos es un problema de investigación alrededor del cual se genera mucha actividad académica.

Lo que se busca es confeccionar un calendario que satisfaga todas las restricciones y condiciones no es una tarea sencilla. Y un calendario preconcebido donde solo haya que asociar etiquetas a equipos no puede tener en cuenta la restricción de minimización de costos de desplazamientos.

Formalización

Dados n equipos (n par) un torneo todos contra todos (ttt) es un torneo en el que cada par de equipos juega un partido entre ellos; y un torneo todos contra todos ida y vuelta (tttiv) es un torneo en el que cada par de equipos juega dos partidos entre ellos: el de ida y el de vuelta

ttt - > consiste de $n - 1$ fechas, cada una conteniendo n/2 partidos.

tttiv -> consiste de $2(n - 1)$ fechas, cada una conteniendo n/2 partidos. En cada juego un equipo se denomina local y el otro visitante. Los dos juegos entre cada par de equipos deben alternar la localía

Representación del calendario:
Ejemplo: Para la matriz m es 2(n-1) y n es 6 equipos.

Si es solo k quiere decir que el equipo j juega de local contra k
Si es -k quiere decir que el equipo j juega de visitante contra k

Propiedades que debe tener el calendario:

- $Cal[i, j] = k$ si y sólo si $Cal[i, k] = -j$
- $\{|Cal[i, j]| : 1 \leq j \leq n\} = \{1, 2, \ldots, n\}$, para todo $1 \leq i \leq 2(n - 1)$.
- $n/2 = |\{Cal[i, j] > 0 : 1 \leq j \leq n\}| = |\{Cal[i, j] < 0 : 1 \leq j \leq n\}|$, para todo $1 \leq i \leq 2(n - 1)$.
- $\forall j \in [1..n], \forall k \neq j, \exists i1, i2 \in [1.,2(n - 1)] : Cal[i1, j] = k \wedge Cal[i2, j] = -k$.

A tener en cuenta:

- Gira: el equipo i juega k (número de partidos) consecutivos de vistinate ante los equipos j
- Permanencia: el equipo i juega k (número de partidos) consecutivos de local ante los equipos j
- Costo: Es la suma de las distancias desde el local a donde inicia la gira + las distancias de la gira y finalmente donde termina la gira hasta el local

¿Entendimos el problema?

Vamos a realizar un ejercicio de entendimiento con base a una permutación realizada en la salida del calendario del ejercicio ejemplo del problema

$$Cal_1 = \begin{bmatrix} 3 & 4 & -1 & -2 \\ 2 & -1 & 4 & -3 \\ -3 & -4 & 1 & 2 \\ 4 & -3 & 2 & -1 \\ -2 & 1 & -4 & 3 \\ -4 & 3 & -2 & 1 \end{bmatrix}$$

Este calendario cumple con todas las restricciones solicitadas.

Como ejercicio vamos a calcular el costo de Cal_1 y comprarlo con el costo del calendario inicial.

El costo de gira del equipo 1 es $d_{13}+ d_{31}$= 1330 y $d_{12}+ d_{24}+ d_{41}$= 2011

El costo de gira del equipo 2 es $d_{21}+ d_{14}+ d_{43}+ d_{32}$= 2134

El costo de gira del equipo 3 es $d_{31}+ d_{13}$= 1330 y $d_{34}+ d_{42}+ d_{23}$= 797

El costo de gira del equipo 4 es $d_{42}+ d_{23}+ d_{34}$= 797 y $d_{41}+ d_{14}$= 1858

El costo total de la solución es 2011 + 1330 + 2134 + 1330 + 797 + 797 + 1858 = 10257

Ahora comparando con el resultado del costo de Cal, se puede notar que el costo total de Cal es 8276 y Cal_1 es 10257, el resultado de Cal es mucho menor que Cal_1 y por eso es que la salida de Cal es la mejor, ya que minimiza el costo de las giras.

También hay permutaciones de Cal que no son soluciones factibles por no cumplir las restricciones como por ejemplo:

$$\begin{bmatrix} 3 & 4 & -1 & -2 \\ -2 & 1 & -4 & 3 \\ 2 & -1 & 4 & -3 \\ -3 & -4 & 1 & 2 \\ 4 & -3 & 2 & -1 \\ -4 & 3 & -2 & 1 \end{bmatrix}$$

No programar el mismo partido en dos fechas consecutivas. Esto se viola en las fechas 2 y 3, y las fechas 5 y 6.

Por último, note también que si en la entrada el valor de Max es 2 y no 3, entonces Cal, Cal_1, Cal_2 no son soluciones factibles (¿por qué?). Como ejercicio de asimilación, encuentre una solución para ese caso.

Solución:

-2	1	4	-3
3	-4	-1	2
4	3	-2	-1
-4	-3	2	1
-3	4	1	-2
2	-1	-4	3

Modelamiento e Implementación

Descripción del modelo genérico:

1. Parámetros
   - n: Número total de equipos en el torneo. $n \geq 0$
   - D : Es una matriz de tamaño n x n, donde la posición $D_{i,j}$ es la distancia entre las ciudades de los equipos **i** y **j**.
   - Min: Tamaño mínimo de la gira o permanencia que puede tener cada equipo j. $1 \leq Min \leq Max$
   - Max: Tamaño máximo de la gira o permanencia que puede tener cada equipo j. $Max \geq Min$

2. Variables

- $Calc_{i,j}$: Contrincante en la fecha i de un equipo j.
  $\forall\, i \in [1, 2 * (n-1)],\ j \in [1, n]$: $Cal[i,j] \geq -n \ \wedge\ Cal[i,j] \leq -n$
- $Giras_{i,j}$: Representa cuántos partidos seguidos lleva seguidos siendo visitante el equipo j en la fecha i.
  $1 \leq j \leq n;\ 1 \leq i \leq 2 * (n-1)$
- $Permanencia_{i,j}$: Representa la cantidad más grande partidos seguidos siendo local el equipo j en la fecha i.
  $1 \leq j \leq n;\ 1 \leq i \leq 2 * (n-1)$

3. Restricciones
   - n es par.

     $n \,\%\, 2 = 0$

   - No programar partidos de vuelta hasta tanto no se hayan programado todos los partidos de ida.

     $\forall$ i, k $\in$ [1, (n-1)], i $\neq$ k, j $\in$ [1, n]: |Cal[i, j]| $\neq$ |Cal[k, j]|

- Hay un partido de ida y uno de vuelta

  $\forall$ j $\in$ [1..n], $\forall$ k $\neq$ j, $\exists$ i1, i2 $\in$ [1.,2(n − 1)] : Cal[i1, j] = k $\wedge$ Cal[i2, j] = −k.

- Que no sea cero ninguna posición en la matriz

  $\forall$ i $\in$ [1,2*(n-1)], j $\in$ [1, n]: Cal [i,j] $\neq$ 0

- No puede repetirse un partido en dos fechas consecutivas

  $\forall$ i $\in$ [1 ,2*(n-1)-1], j $\in$ [1, n]: |Cal [i,j] | $\neq$ |Cal [i+1,j] |

- Equipo local en una fecha y el otro juega como visitante

  $\forall\, i \in [1, 2\ *\ (n\ -\ 1)],\ j, k \in [1,\ n]: (Cal[i, j]\ =\ k) \leftrightarrow (Cal[i,\ k]\ =\ \ -\ j)$

- No se pueden repetir partidos.

  $\forall\, i, k \in [1, 2\ *\ (n\ -\ 1)], \forall i \neq\ k,\ j \in [1,\ n]:\ Cal[i, j] \neq Cal[k, j]$

- Todo número en la matriz Gira debe ser menor o igual que Max mayor o igual que 0.
  $\forall\, i \in [1,\ \dots\ ,\ 2\ *\ (n\ -\ 1)]\ ,\ j \in [1,\ \dots\ ,\ n]\ \rightarrow\ Giras_{i,\,j} \leq Max \wedge Giras_{i,\,j} \geq 0$

- Todo número en la matriz Permanencia debe ser menor o igual que Max y mayor o igual que 0.

$$\forall\, i \in [1,\ \dots\ ,\ 2\ *\ (n\ -\ 1)]\ ,\ j \in [1,\ \dots\ ,\ n]\ \rightarrow\ Permanencia_{i,\,j} \leq Max \wedge Permanencia_{i,\,j} \geq 0$$

- Restricción para llenar la matriz de gira:

$$\forall\, i \in [1,\ \dots\ ,\ 2\ *\ (n\ -\ 1)]\ ,\ j \in [1,\ \dots\ ,\ n] \rightarrow$$
$$[\ Cal_{i,\,j} < 0 \rightarrow (\ (i\ =\ 1 \rightarrow Giras_{i,\,j} = 1)\ \wedge (i \neq 1 \rightarrow Gira_{i,\,j} = Gira_{i-1,\,j} + 1)]$$
$$\wedge$$
$$[\ Cal_{i,\,j} > 0 \rightarrow Gira_{i,\,j} = 0]$$

- Restricción para llenar la matriz de permanencia:

$$\forall\, i \in [1,\ \dots\ ,\ 2\ *\ (n\ -\ 1)]\ ,\ j \in [1,\ \dots\ ,\ n] \rightarrow$$
$$[\ Cal_{i,\,j} > 0 \rightarrow (\ (i\ =\ 1 \rightarrow Permanencia_{i,\,j} = 1)\ \wedge (i \neq 1 \rightarrow Permanencia_{i,\,j} = Permanencia_{i-1,\,j} + 1)]$$
$$\wedge$$
$$[\ Cal_{i,\,j} < 0 \rightarrow Permanencia_{i,\,j} = 0]$$

- Restricción para mantener un número min de giras

$$\forall\, i \in [1, \dots, 2 * (n-1)]\,,\ j \in [1, \dots, n] \rightarrow$$
$$(i = 2 * (n-1)) \rightarrow [Cal_{i,j} < 0 \rightarrow Giras_{i,j} \geq Min]$$
$$\vee$$
$$(i = 1) \rightarrow [Cal_{i,j} < 0 \wedge Cal_{i+1,j} > 0] \rightarrow Giras_{i,j} \geq Min$$
$$\vee$$
$$\forall\, i \neq [1 \vee 2 * (n-1)] \rightarrow [Cal_{i,j} > 0 \wedge Cal_{i-1,j} < 0] \rightarrow Giras_{i-1,j} \geq Min$$

- Restricción para mantener un número min de permanencias

$$\forall\, i \in [1, \dots, 2 * (n-1)]\,,\ j \in [1, \dots, n] \rightarrow$$
$$(i = 2 * (n-1)) \rightarrow [Cal_{i,j} > 0 \rightarrow Permanencia_{i,j} \geq Min]$$
$$\vee$$
$$(i = 1) \rightarrow [Cal_{i,j} > 0 \wedge Cal_{i+1,j} < 0] \rightarrow Permanencia_{i,j} \geq Min$$
$$\vee$$
$$\forall\, i \neq [1 \vee 2 * (n-1)] \rightarrow [Cal_{i,j} < 0 \wedge Cal_{i-1,j} > 0] \rightarrow Permanencia_{i-1,j} \geq Min$$

4. Función objetivo

- Minimizar el costo total de las distancias con las permanencias y las giras

$$\sum_{i=1}^{2*(n-1)} \sum_{j=1}^{n}$$
$$0 \quad si \quad Cal[i,\ j] > 0 \quad \wedge \quad i = 1\,,$$
$$D[j,\ |Cal[i,\ j]|\ si \quad i = 1 \quad \wedge \quad Cal[i,\ j] < 0\,,$$
$$D[|Cal[i-1,\ j]|,\ |Cal[i,\ j]|] + D[|Cal[i,\ j]|,\ j]\ si\ Cal[i,\ j] < 0 \wedge i = 2 * (n-1) \wedge Cal[i-1,\ j] < 0,$$
$$D[|Cal[i,\ j]|,\ j] + D[|Cal[i,\ j]|,\ j]\ si\ Cal[i,\ j] < 0 \wedge i = 2 * (n-1) \wedge Cal[i-1,\ j] > 0,$$
$$D[|Cal[i-1,\ j]|,\ |Cal[i,\ j]|]\ si\ Cal[i,\ j] < 0 \wedge Cal[i-1,\ j] < 0,$$
$$D[|Cal[i,\ j]|,\ j]\ si\ Cal[i,\ j] < 0 \wedge Cal[i-1,\ j] > 0,$$
$$D[j,\ |Cal[i-1,\ j]|\ si\ Cal[i,\ j] > 0 \wedge Cal[i-1,\ j] < 0,$$
$$0\ si\ Cal[i,\ j] > 0 \wedge Cal[i-1,\ j] > 0$$

Primera fecha.
Última fecha.
Cualquier otra fecha.

Detalles importantes de implementación

Para la realización de las restricciones se tuvieron en cuenta los siguientes aspectos:

- El número de equipos es par.
- Un partido enfrenta a dos equipos, uno de ellos es local y el otro visitante.
- Una fecha es un conjunto de partidos donde cada equipo juega una y solo una vez.
- Un calendario a una ronda es una secuencia de fechas tal que en el conjunto de todos los partidos cada equipo se enfrenta una sola vez a cada otro equipo.
- Un calendario a dos rondas es un conjunto de fechas tal que en el conjunto de todos los partidos cada equipo se enfrenta exactamente dos veces a cada otro equipo; en estos dos enfrentamientos los equipos se alternan la localía. Al primer partido entre dos equipos se le llama el partido de ida y al segundo se le llama el partido de vuelta.
- No programar partidos de vuelta hasta tanto no se hayan programado todos los partidos de ida.

Nuestro modelo busca minimizar el costo total de desplazamientos, y esto podría variar en base a las restricciones tenidas en cuenta. Por ejemplo, si se permitiera el enfrentamiento entre dos equipos dos fechas seguidas, o que se puedan programar partidos de vuelta antes de jugar los de ida, podrían disminuir o aumentar los costos, sin embargo esto haría que aumentaran en gran medida las posibles soluciones, generando dificultad para que el programa terminase su ejecución.

Análisis de árboles generados (branch and bound)

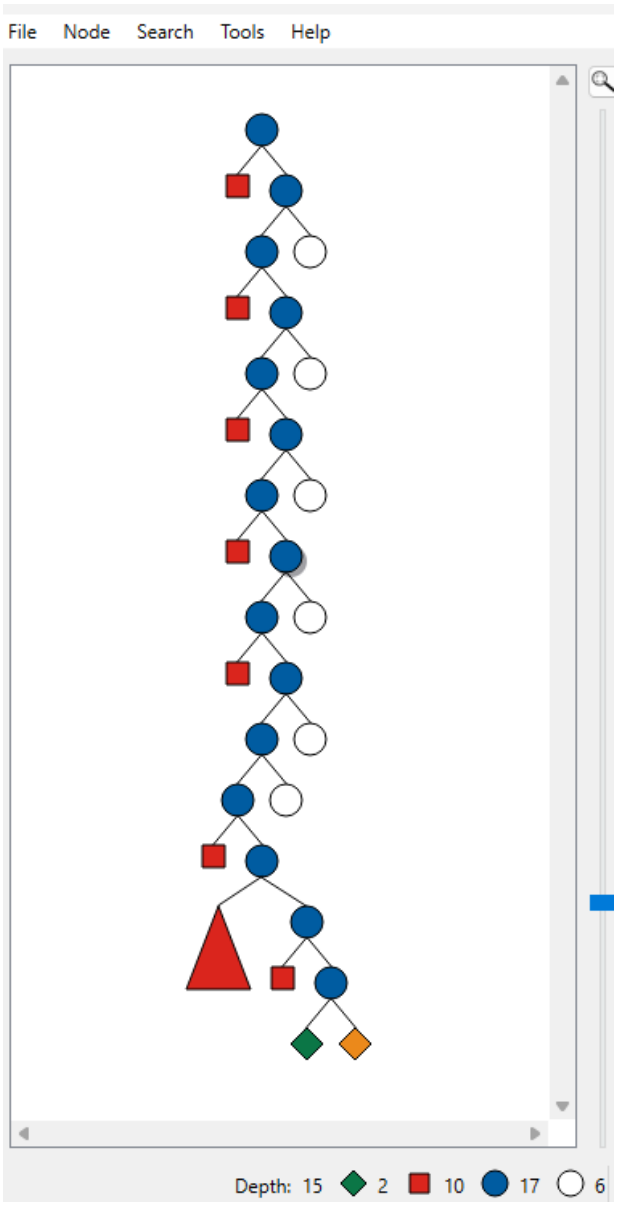


Imagen 1

En la imagen anterior podemos notar que tiene de profundidad 15, 10 ramas con una solución no viable, 17 nodos explorados y 6 nodos sin explorar, también se puede notar que al final se logra llegar a dos tipos de soluciones la primera es en la rama de color verde donde se puede representar una solución optima, este puede ser considerado el mejor valor encontrado hasta el momento y también es potencial candidato a solución óptima. La segunda solución está en proceso de exploración y/o análisis.

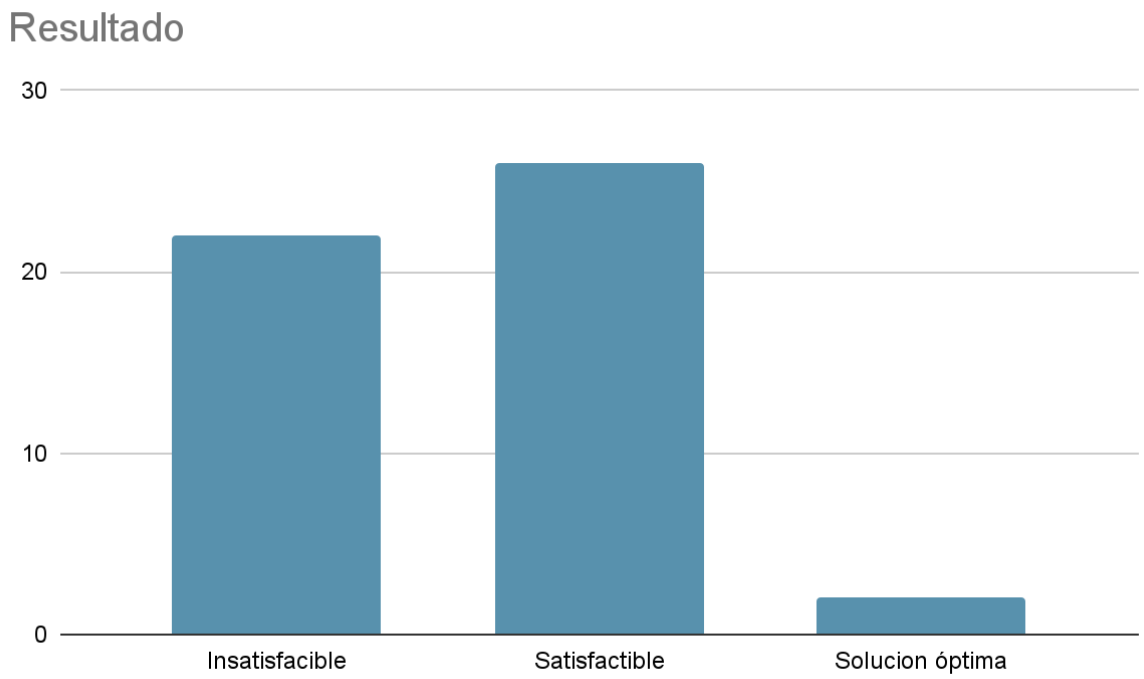
Casos de pruebas

Prueba	Costo	Tipo de solución	Tiempo de ejecución
case4_0	None	Insatisfacible	0.424239 s
case4_1	1984	Satisfactible	0.374128 s
case4_2	6184	Satisfactible	0.470860 s
case4_3	6224	Satisfactible	0.467252 s
case4_4	4612	Solución Óptima	0.471125 s
case4_5	None	Insatisfacible	0.339436 s
case4_6	None	Insatisfacible	0.362970 s
case4_7	5736	Satisfactible	0.494043 s
case4_8	None	Insatisfacible	0.389647 s
case6_0	None	Insatisfacible	0.850247 s
case6_1	None	Insatisfacible	1.577406 s
case6_2	13192	Satisfecho	2 min 0.611969 s
case6_3	None	Insatisfacible	02 min 477966 s
case6_4	12074	Satisfactible	02 min 00.842532 s
case6_5	16216	Satisfactible	02 min 01.930726 s
case6_6	8086	Satisfactible	02 min 00.840428 s
case6_7	13302	Satisfactible	02 min 01.257756 s
case6_8	None	Insatisfacible	01.966155 s
case8_0	None	Insatisfacible	02 min 682100 s
case8_1	26174	Satisfactible	02 min 01.507565 s
case8_2	33532	Satisfactible	02 min 01.603758 s
case8_3	21148	Satisfactible	02 min 01.643420 s
case8_4	20460	Solución óptima	02 min 01.486568 s

case8_5	29334	Satisfactible	02 min 01.531780 s
case8_6	25848	Satisfactible	02 min 01.595817 s
case8_7	20642	Satisfactible	02 min 01.601269 s
case8_8	23878	Satisfactible	02 min 01.690222 s
case10_0	None	Insatisfacible	3.584184 s
case10_1	34134	Insatisfacible	2 min 1.775631 s
case10_2	41750	Satisfactible	2 min 1.766638 s
case10_3	29244	Satisfactible	2 min 01.751579 s
case10_4	38084	Satisfactible	2 min 1.745326 s
case10_5	37820	Satisfactible	2 min 1.709659 s
case10_6	39304	Satisfactible	2 min 1.770543 s
case10_7	45662	Satisfactible	2 min 1.788082 s
case10_8	36740	Satisfactible	2 min 1.781786 s
case12_0	None	Insatisfacible	6.538273 s
case12_1	53318	Satisfactible	2 min 2.424414 s
case12_2	52922	Satisfactible	2 min 2.508908 s
case12_3	75852	Satisfactible	2 min 2.451675 s
case12_4	None	Insatisfacible	2 min 2.319181 s
case16_0	None	Insatisfacible	17.805372 s
case16_1	None	Insatisfacible	2 min 5.253091 s
case16_2	None	Insatisfacible	2 min 5.148286 s
case16_3	None	Insatisfacible	2 min 5.108395 s
case16_4	None	Insatisfacible	2 min 5.643519 s
case20_0	None	Insatisfacible	2 min 17.058140s
case20_1	None	Insatisfacible	2 min 17.007191s

case20_2	None	Insatisfacible	2 min 17.007191
case20_3	None	Insatisfacible	2 min 17.007191
Pruebas propias			
Caso instancia propia 1	14367	Satisfactible	5 minutos
Caso instancia propia 2	5478	Solución óptima	400msec
Caso instancia propia 3	12923	Satisfactible	5 minutos
Caso instancia propia 4	3974	Solución óptima	424msec
Caso instancia propia 5	12686	Satisfactible	43 minutos

Análisis de pruebas



Como vemos, nuestro modelo tuvo una eficiencia en cuanto a encontrar una solución del 56%, que puede ser bastante bajo teniendo en cuenta que el modelo implementado por el profesor fue del 88%. Sin embargo esto se debe plenamente a las restricciones propuestas en

el lenguaje formal, ya que de tener más, menos o diferentes restricciones el resultado podría variar, para muestra la restricción de jugar el mismo partido dos jornadas seguidas.
Los resultados podrían variar si se da más tiempo de ejecución a este, ya que está limitado a apenas dos minutos para la batería de pruebas.

En cuanto a las instancias propias, podemos evidenciar lo hablado en que de dejar más tiempo el programa corriendo este podría encontrar una solución como en el caso de la instancia 5.

Vídeo

En el siguiente link se encuentra el video donde se explica todo acerca de nuestra interfaz gráfica -> https://youtu.be/fgYZoNeky0o

Repositorio

En este link se encuentra el repositorio github del proyecto -> https://github.com/mayra-Sanchez/ADA2_proyecto2

Conclusiones

El modelo es capaz de proponer soluciones que cumplan con las restricciones propuestas, sin embargo es bastante sensible al cambio en los parámetros de Max y Min, dado esto es bastante probable que el programa falle cuando se trate de alternar entre local y visitante en todos los encuentros.

Resolver el problema de agendamiento de partidos no es algo sencillo, y seguramente para esto se tienen en cuenta muchos más factores además de las distancias, como los costos de viajes dependiendo las ciudades, las ganancias previstas para cierto tipo de encuentro, las condiciones climatológicas y el estadio para donde está previsto un encuentro, entre otras que pueden facilitar mucho la toma de decisiones en cuanto a los partidos, aunque esto de nuevo posiblemente no va a satisfacer al completo lo que se busca, es decir que la solución sea la de menor distancia recorrida.

Para hacer que el modelo pudiese proponer más frecuentemente una solución (aunque esto no garantiza que sea la de menor costo a comparación con el modelo actual), se puede poner una restricción más definida para que siempre se alternen los partidos entre local y visitante, ya que esa restricción es de las que más falla en nuestro modelo, generando que por ejemplo para 4 ciudades, se necesite que el Min sea 1 y el Max 3 para brindar una solución.

En conclusión, el modelo en minizinc cuenta con las restricciones que propusimos, sin embargo estas restricciones podrían ser mejorables, y quizás conociendo un poco más el lenguaje y lo que permite, proponer restricciones que nos ayuden a cumplir con el objetivo, por ejemplo, ajustando la restricción de que no se puede repetir un partido dos fechas seguidas, ya que esto en ocasiones genera que el problema sea insatisfactible.