UNIVERSIDAD SIMÓN BOLÍVAR

CI5437

Informe Proyecto #3

Estudiantes:

Pietro Iaia 15-10718

Diego Peña 15-11095

Sartenejas, Julio 2021

1- Introducción

Los torneos son eventos muy comunes en la actualidad. Todos los días, en algún lugar del mundo se juega alguno: De e-sports, de tenis, de béisbol, etc. Uno de los formatos más populares, usado por la mayoría de las ligas de fútbol europeo, donde cada equipo se enfrenta dos veces contra cada uno de sus rivales. Una como local y otra como visitante.

Para diseñar calendarios para este tipo de competencias, en los cuales se garantice que el mismo cumpla con ciertas condiciones impuestas por el organizador, como fechas, horas, número de partidos, etc. El uso de software puede facilitar mucho el proceso. Aunque existen muchas maneras de hacer esto, una forma que se puede implementar de manera relativamente sencilla, es reducir el problema a una instancia de SAT, llevando las condiciones impuestas por la organización del torneo a cláusulas booleanas que puedan ser resueltas por un solver especializado.

El objetivo de este proyecto es formular e implementar una reducción eficiente a SAT del problema de organización de torneos, dadas ciertas condiciones que se imponen para realizar la misma tarea.

2- Reducción del problema a SAT

De acuerdo al enunciado suministrado, se debe reducir el problema de reducción de organización de torneos a SAT tomando en cuenta las siguientes condiciones:

- Todos los participantes deben jugar dos veces con cada uno de los otros participantes, una como "visitantes" y la otra como "locales".
- Dos juegos no pueden ocurrir al mismo tiempo.
- Un participante puede jugar a lo sumo una vez por día.
- Un participante no puede jugar de "visitante" en dos días consecutivos, ni de "local" dos días seguidos.
- Todos los juegos deben empezar en horas "en punto”.
- Todos los juegos deben ocurrir entre una fecha inicial y una fecha final especificadas. Pueden ocurrir juegos en dichas fechas.

- Todos los juegos deben ocurrir entre un rango de horas especificado, el cual será fijo para todos los días del torneo.
- A efectos prácticos, todos los juegos tienen una duración de dos horas.

En la interpretación utilizada en este informe, se tiene que los partidos pueden ocurrir en el último día, pero que el último partido debe terminar antes de la última hora disponible.

Para llevar el problema a SAT primero se deben definir las variables a utilizar. Es por ellos que se definen dos categorías de variables booleanas que representan los partidos:

- x_{ijd}: Donde i es el equipo local, j el equipo visitante y d el día del torneo. Además $i \neq j$ ya que es imposible que un equipo juegue contra sí mismo.
- y_{ijh}: Donde i es el equipo local, j el equipo visitante y h la hora del partido. Además $i \neq j$ ya que es imposible que un equipo juegue contra sí mismo.

Para representar todas las posibles combinaciones de partidos en cada día/hora posible, se requerirá entonces el número de variables dado por la siguiente expresión:

$$P * (D + H)$$

Donde:

- P es el número de partidos jugados, obtenido mediante la expresión $P = N * (N - 1)$ donde N es el número de partidos.
- D es el número de días que dura el torneo.
- H es el número de horas disponibles para jugar en un solo día, o visto de otra forma, el número máximo de juegos que pueden ocurrir en un día.

Dado que con las condiciones de duración de los juegos y del número máximo de juegos en un día, se tiene que un torneo que se juegue todas las 24 horas del día, puede tener un máximo de 12 juegos por día. Luego, este factor puede no resultar tan influyente en la cantidad de variables, ya que los otros dos: Equipos y duración en días no tienen una cota superior.

En cuanto a las cláusulas de SAT que se requieren para resolver el problema, es posible agruparlas de acuerdo a la función que cumplen. Nota: A partir de ahora, cuando se haga referencia al partido (i, j), se hace referencia al partido i contra j donde i es local. Por lo tanto (i, j) no es lo mismo que (j, i) y por ello se utiliza notación de pares ordenados para representar la información.

Antes de enumerar las restricciones, es importante explicar la interpretación que se le da al conjunto *Horas.* Como cada partido dura dos horas, debe empezar en una hora en punto y no pueden ocurrir enfrentamientos simultáneamente, se divide el día en bloques de dos horas a partir de la hora inicial, y el último bloque sería dos horas antes del final. Luego *Horas* representa el conjunto de estos bloques. Esto impide que se den ciertos resultados, por ejemplo, que un día determinado, un juego empiece a las 10:00 y otro a la 1. Sin embargo, es imposible que esta decisión vuelva un problema SAT a uno UNSAT ya que:

a) Si un juego es a las 10:00 am y otro a las 1:00 pm), y como es imposible que haya alguien jugando a las 12:00 m (Chocaría con el juego de la 1:00) se puede adelantar el juego (Y cualquier juego posterior en ese mismo día) una hora se tendrían los mismos juegos en el mismo día, solo que a horas distintas. Inclusive, en el horario original sobraba una hora al final, ahora sobrarán dos, y se podría agendar un partido más en ese día si hiciera falta.

b) Si se organizan los juegos en bloques de dos horas a partir de la hora inicial, es imposible que queden bloques libres de una hora entre juego y juego. Por lo tanto, se maximiza el número de juegos posibles en un día.

c) Con esta interpretación, no hacen falta cláusulas adicionales para modelar el caso donde la hora final de un juego coincide con la primera hora de otro, lo cual disminuye la complejidad del problema, basta con verificar que no ocurran en el mismo bloque.

Una vez explicados estos detalles, se procede a enumerar los distintos tipos de cláusulas presentes en una instancia cualquiera del problema:

- Cada partido (i, j) se debe jugar al menos una vez: Sean i, j equipos arbitrarios:
  - $(\exists\, d \mid d \in Días : x_{ijd})$

- $(\exists\ h \mid h \in Horas : y_{ijh})$
- En total se tendrán $P\ *\ (D\ +\ H)$ cláusulas de este estilo.

- Cada partido (i, j) se debe jugar máximo una vez: Sean i, j equipos arbitrarios:
  - $(\forall\ d_1,\ d_2 \mid d_1,\ d_2 \in Días \wedge\ d_1 \neq d_2 : \neg x_{ijd_1} \vee\ \neg x_{ijd_2})$
  - $(\forall\ h_1,\ h_2 \mid h_1,\ h_2 \in Horas \wedge\ h_1 \neq h_2 : \neg y_{ijh_1} \vee\ \neg y_{ijh_2})$
  - En total se tendrán $P\ *\ \frac{D*(D-1)\ +H*(H-1)}{2}$ cláusulas de esta forma
- No se permiten juegos simultáneos: Sean los partidos (i, j) y (k, m) tal que $(i,\ j) \neq (k,\ m)$:
  - La forma instintiva de modelar esta restricción sería $(\forall\ d,\ h \mid d \in Días \wedge\ h \in Horas : x_{ijd} \wedge\ x_{kmd} \Longrightarrow \neg y_{ijh} \vee\ \neg y_{kmh})$ sin embargo, esto no está en CNF. Luego, mediante uso de lógica simbólica, es posible generar la siguiente restricción equivalente $(\forall\ d,\ h \mid d \in Días \wedge\ h \in Horas : \neg x_{ijd} \vee\ \neg x_{kmd} \vee\ \neg y_{ijh} \vee\ \neg y_{kmh})$
  - En total se tendrán $\binom{P}{2} *\ D\ *\ H$ cláusulas de este estilo.
- Cada equipo solo puede jugar una vez por día: Sean $i \in Equipos$ y $d \in Días$ arbitrarios, se tienen:
  - $(\forall\ j,\ k \mid j, k \in Equipos \wedge\ i \neq j \wedge\ i \neq k \wedge\ j \neq k : \left(\neg x_{ijd} \vee\ \neg x_{ikd}\right) \wedge \left(\neg x_{ijd} \vee\ \neg x_{kid}\right) \wedge\ \left(\neg x_{jid} \vee\ \neg x_{ikd}\right) \wedge\ (\neg x_{jid} \vee\ \neg x_{kid}))$
  - $(\forall\ j \mid j \in Equipos : \neg x_{ijd} \vee\ \neg x_{jid})$
  - En total se tendrán $4\ *\ N\ *\ D\ *\ \binom{N-1}{2} + D\ *\ \left(\frac{N*(N-1)}{2}\right)$ cláusulas de este estilo.
- Un equipo no puede jugar como local en días consecutivos, ni como visitante: Sea $i \in Equipos$ arbitrario:
  - $(\forall\ j,\ k \mid j,\ k \in Equipos \wedge\ d \in Días \wedge\ \neg ÚltimoDía(d) \wedge\ i \neq j \wedge\ i \neq k \wedge\ j \neq k : \left(\neg x_{ikd} \vee\ \neg x_{ijd+1}\right) \wedge\ \left(\neg x_{ijd} \vee\ \neg x_{ikd+1}\right) \wedge\ \left(\neg x_{kid} \vee\ \neg x_{jid+1}\right) \wedge\ \left(\neg x_{jid} \vee\ \neg x_{kid+1}\right))$
  - En total se tendrán $4\ *\ N\ *\ \binom{N-1}{2} *\ (D\ -\ 1)$ cláusulas de este estilo.

3- Implementación

Parte 1: Reducción a SAT

Para implementar la reducción se utilizó un script de Python que recibe como entrada un archivo JSON con el siguiente formato, indicado por el profesor, y produce un archivo txt con las cláusulas correspondientes en el formato DIMACS CNF.

{
"tournament_name": String. Nombre del torneo,
"start_date": String. Fecha de inicio del torneo en formato ISO 8601,
"end_date": String. Fecha de fin del torneo en formato ISO 8601,
"start_time": String. Hora a partir de la cual pueden ocurrir los juegos en cada día, en formato ISO 8601,
"end_time": String. Hora hasta la cual pueden ocurrir los juegos en cada día, en formato ISO 8601,
"participants": [String]. Lista con los nombres de los participantes en el torneo
}

El programa agrupa las restricciones que debe escribir en el archivo de acuerdo a las categorías descritas en la página anterior. Si la hora de inicio no es en punto, se redondea "hacia arriba", es decir, a la hora en punto más próxima y que ocurra después de la hora señalada, ya que de acuerdo a las normas esa será la primera hora a la que puede empezar un torneo. Algo similar ocurre con la hora de fin, pero redondeando la hora "hacia abajo". El archivo generado se llama SAT_<Nombre-del-torneo>.txt pero con los espacios en el nombre reemplazados por un guión.

Pasar de las variables utilizadas a DIMACS

El formato DIMACS requiere que se numeren las variables del 1 al máximo de variables utilizadas. Sin embargo, como se pudo ver anteriormente, el modelado propuesto usa dos tipos de variables, cada una con tres índices, así que debe existir una biyección que permita traducir estos en las variables vistas anteriormente. Para este proyecto se utilizaron dos funciones, una para cada tipo de variable:

- Sea la variable x_{ijd}con $i, j \in Equipos, i \neq j \, y \, d \in Días$ se tiene la función $f_1(i, j, d) = ((N - 1) + D)i + Dj + d$. Para poder llevar esto a cabo es necesario identificar los equipos con números del 0 al N – 1, para lo cual se utiliza el índice que tienen en el arreglo que contiene el JSON, mientras que los días se identifican del 1 al número de días.
- Sea la variable x_{ijh}con $i, j \in Equipos, i \neq j \, y \, h \in Horas$ se tiene la función $f_2(i, j, h) = ((N - 1) + H)i + Hj + h + maxd2n$. El valor de *maxd2n* es el máximo valor que se puede obtener con f_1 en el dominio de las variables de ese problema. Es decir: $maxd2n = f_1(N - 1, N - 1, D)$. Esto impide que los valores de $f_1$coincidan con los de f_2: Los mayores que *maxd2n* pertenecerán a las variables que representan la hora de un partido y que utilizan f_2, mientras que el resto pertenecerá al grupo de variables que representa la fecha del partido y utiliza f_1.

Para obtener los índices a partir del número, en primer lugar se debe determinar si el número es mayor o igual a maxd2n, con la intención de saber si se debe usar la inversa de f_1 o la inversa de f_2. Luego, se puede aplicar el siguiente procedimiento estándar para recuperar los valores de los tres índices:

```
def n_to_triple(number, coef, teams, fterm):
    if number % coef == 0:
        local = (number // coef) - 1
        return local, teams - 1, fterm
    else:
        local = number // coef
        rest = number % coef
        if rest % fterm == 0:
            visitor = (rest // fterm) - 1
            return local, visitor, fterm
        else:
            visitor = rest // fterm
            return local, visitor, rest % fterm
```

Imagen 1: Procedimiento para recuperar la información de una variable a partir de un entero positivo

Donde number es el número que representa a la variable, coef es el coeficiente por el cual se multiplica al primer parámetro en una función f, teams es el número de equipos y fterm es el coeficiente del segundo parámetro. Nótese que para aplicar este procedimiento sobre un número mayor que maxd2n, se debe restar maxd2n sobre el mismo para que dé los resultados correctos.

Una desventaja de estas funciones, es que el máximo factible (es decir, el valor máximo que retorna la función dada una variable que aparezca en una cláusula, ya que variables de tipo X_{iid} o Y_{iih}no aparecen en ninguna) es mucho mayor al máximo número de variables que hay en cada categoría. Esto se debe a que para efectos de la función, las variables cuyos dos primeros índices son iguales, o en otras palabras, que representan enfrentamientos entre el mismo equipo, sí existen, aunque jamás se usen en ninguna cláusula. Esto hace que el número de variables que requiere el solver aumente.

Parte 2: Resolución del SAT

El archivo .txt generado en la etapa anterior se pasa como input a un SAT solver. Para este proyecto se utilizó Glucose 4.1, versión no paralelizada. Esto solver fue diseñado e implementado por Gilles Audemard and Laurent Simon, por primera vez en 2009. La versión actual es de 2015-2016. Está inspirado en un solver conocido como Minisat, creado por Niklas Eén y Niklas Sörensson. El archivo de output tiene el nombre SOL_<nombre-del-torneo>.txt con los espacios cambiados por nombres.

Parte 3: Generación del calendario

Se utiliza un script de Python para pasar el resultado de SAT a ical. Se utiliza la librería icalendar para Python. En caso de que Glucose le haya asignado valor verdadero a un número que representa una variable que no está en ninguna cláusula (Cosa que no afecta el resultado final), es decir a una variable que representa un partido de un equipo contra sí mismo, estas variables son ignoradas a la hora de generar el calendario. De nuevo, esto no afecta el resultado porque esas variables no aparecían en ninguna cláusula y no representaban ningún partido real. El archivo ical tiene el nombre del torneo.

NOTA IMPORTANTE: Los calendarios se están generando con las horas en UTC. Algunos manejadores de calendarios hacen un ajuste automático a la zona horaria correspondiente al computador (Por ejemplo la aplicación de calendarios de Ubuntu) por lo que se recomienda poner su computador la zona horaria UTC antes de abrir un calendario, especialmente si ve que alguna de las restricciones no se está cumpliendo, ya que al hacer ajustes horarios, partidos cercanos a la medianoche podrían cambiar de día. Esto no es un error de ninguno de los scripts de Python, sino de cómo los manejadores de calendarios trabajan los horarios en los archivos.

4- Ejecución:

- Para generar casos de prueba: Se debe ejecutar el script de Python:
  python3 generator.py <# de casos a generar> <# de participantes> <# tipo de torneo>
- Para generar un calendario: Se debe ejecutar el script de Python (Requiere versión 3.6 en adelante): python3 newCalendar.py <nombre del archivo JSON>
- Para fines experimentales, se creó el script testResults.py, el cual ejecuta el proceso sobre todos los archivos JSON que se encuentren en un mismo directorio. Durante su ejecución este genera tres archivos, donde cada uno guarda los tiempos de una etapa distinta del proceso: Reducción a SAT, resolución de SAT y creación de calendarios.

5- Casos de Prueba:

Para poder casos de prueba lo suficientemente diferentes, se decidió dividir los torneos en 3 tipos: Torneos relajados, normales y ajustados, donde básicamente cada uno determina qué tan ajustado, en número de días, se encuentran dichos torneos. El proceso, que se llevó a cabo para generar estos casos de prueba, primero calcula el número mínimo de horas que debe durar el torneo para que todos los partidos puedan ser jugados, luego toma esta información, y establece el horario diario y la cantidad de días que dura el torneo.

Para los torneos relajados, estos pueden durar desde una semana hasta 18 días más que el número mínimo de horas, es decir que si el torneo necesita 1 semana mínima, los torneos relajados podrán durar entre 2 semanas o 3 semanas y media. Para los torneos

normales, estos pueden durar desde 3 días hasta 10 días más que el número mínimo de horas. Por último, para los torneos ajustados, estos pueden durar desde el número mínimo de horas hasta 2 días más que el número mínimo de horas.

6- Recursos:

- Laptop con Intel Core i5 2.30GHz.

7- Resultados:

# de Participantes	Torneo relajado	Torneo normal	Torneo ajustado	Todos
10	80%	60%	60%	66%
15	-	40%	-	40%
20	80%	50%	30%	53%
Todos	80%	50%	45%	57%

Tabla 7.1: Porcentaje de casos resueltos (SAT) por cada tipo de torneo.

# de Participantes	Torneo relajado	Torneo normal	Torneo ajustado	Todos
10	10%	30%	30%	23%
15	-	0%	-	0%
20	0%	0%	0%	0%
Todos	5%	10%	15%	10%

Tabla 7.2: Porcentaje de casos no resueltos (UNSAT) por cada tipo de torneo.

# de Participantes	Torneo relajado	Torneo normal	Torneo ajustado	Todos
10	0.0102	0.0099	0.0099	0.0100
15	-	0.0236	-	0.0236
20	0.0453	0.0641	0.052	0.0524
Todos	0.0277	0.0316	0.0239	0.0283

Tabla 7.3: Tiempo promedio (en segundos) de generación de archivo calendario por cada tipo de torneo.

# de Participantes	Torneo relajado	Torneo normal	Torneo ajustado	Todos
10	495	364	364	416
15	-	57103	-	57103
20	22935	172876	70259	78665
Todos	11715	72998	23662	37384

Tabla 7.4: Porcentaje de conflictos, en casos resueltos (SAT), por cada tipo de torneo.

# de Participantes	Torneo relajado	Torneo normal	Torneo ajustado
10	359319	25131	25131

Tabla 7.5: Porcentaje de conflictos, en casos no resueltos (UNSAT), por cada tipo de torneo.

# de Participantes	Torneo relajado	Torneo normal	Torneo ajustado	Todos
10	2804	2565	2565	2661
15	-	239937	-	239937
20	139501	516864	255998	279273
Todos	71152	237297	87043	137032

Tabla 7.6: Porcentaje de decisiones, en casos resueltos (SAT), por cada tipo de torneo.

# de Participantes	Torneo relajado	Torneo normal	Torneo ajustado
10	435882	33215	33215

Tabla 7.7: Porcentaje de decisiones, en casos no resueltos (UNSAT), por cada tipo de torneo.

# de Participantes	Torneo relajado	Torneo normal	Torneo ajustado	Todos
10	4.939	2.9955	2.9955	3.7729
15	-	6.8750	-	6.8750
20	15.5860	72.8917	25.2725	35.3102
Todos	10.2625	27.3287	10.4211	16.6980

Tabla 7.8: Tiempo promedio (en segundos) en encontrar solución (SAT) para cada tipo de torneo.

# de Participantes	Torneo relajado	Torneo normal	Torneo ajustado
10	26.7624	4.0827	4.0827

Tabla 7.9: Tiempo promedio (en segundos) en encontrar que no tiene solución (UNSAT) para cada tipo de torneo.

- Restricción #1: Cada partido debe ocurrir al menos una vez.
- Restricción #2: Cada partido debe ocurrir máximo una vez
- Restricción #3: Dos juegos no pueden ocurrir al mismo tiempo.
- Restricción #4: Un participante puede jugar a lo sumo una vez por día.
- Restricción #5: Un participante no puede jugar de "visitante" en dos días consecutivos, ni de "local" dos días seguidos.

Nota: Entre la restricción #1 y #2, se garantiza que cada partido posible ocurra una sola vez por torneo, es decir, garantiza que cada participante juega una sola vez como visitante y una sola vez como anfitrión.

# de Participantes	Pre-cálculo	Restricción #1	Restricción #2	Restricción #3	Restricción #4	Restricción #5	Todos
10	0	0.0019	0.0414	2.2842	0.0592	0.062	0.4081
15	0	0.0071	0.2600	20.6518	0.3196	0.3084	3.5911
20	0.0001	0.0227	1.4326	101.9285	1.3207	1.3422	17.6744
Todos	0	0.0115	0.6688	47.6128	0.6370	0.6458	8.2626

Tabla 7.10: Tiempo promedio (en segundos) en calcular las restricciones para cada tipo de torneo.

# de Participantes	Restricción #1	Restricción #2	Restricción #3	Restricción #4	Restricción #5	Todos
10	180	22860	506232	29848	27054	117234
15	420	132657	5376525	185871	176904	1174475
20	760	932381	29418485	886755	860928	6419862
Todos	463	428340	13592954	419383	405836	2969395

Tabla 7.11: Número promedio de cláusulas calculadas por restricciones para cada tipo de torneo.

8- Análisis de resultados:

Antes de empezar el análisis, debemos conocer el porcentaje de soluciones que fueron encontradas, y las que no, en la corrida de nuestros casos de prueba. Esta

información puede ser encontrada en las tablas 7.1 y 7.2 respectivamente. Podemos notar también que si juntamos los porcentajes, para la mayoría de los tipos de casos de prueba no llegaremos al 100% de casos, ya que el porcentaje restante de casos, es decir los que no dieron SAT ni UNSAT, son los casos en donde no se logró encontrar una solución, o en su defecto encontrar que no tenían solución, dentro del límite de tiempo que se impuso (TIMEOUT = 10 minutos).

Para los casos de prueba con 15 participantes en torneos ajustados y en torneos relajados, no pudimos llegar a encontrar soluciones ya que tardaban demasiado tiempo, y decidimos no incluirlos en el informe. En total se tomaron en cuenta 70 casos de prueba, 10 por cada tipo de torneo que se tomó en cuenta.

Durante la fase de implementación del proyecto, nos dimos cuenta de que si la función que transforma las variables SAT a enteros hubiera tenido un máximo más ajustado al número total de variables, el solver hubiera tenido que tomar en cuenta menos variables y posiblemente se hubiera tardado menos.

En las tablas 7.1 y 7.2 podemos notar que a medida que los torneos se vuelven más ajustados en tiempo, el porcentaje de solución (SAT) decrementa, y el porcentaje de no solución (UNSAT) incrementa. El hecho de que uno incremente y el otro decremente es independiente al tipo de torneo, ya que depende de si el caso es verdaderamente solucionable o no, lo que verdaderamente depende del tipo de torneo es si llegan a TIMEOUT. Esto sucede ya que mientras más ajustado esté en tiempo, más conflictos se generan, y el algoritmo tarda más en devolverse por backtracking y probar por una rama diferente. La cantidad de participantes también influye en estos resultados, ya que mientras más participantes, menos porcentaje de solución obtenemos.

En la tabla 7.3 podemos notar que para los torneos con mayor cantidad de participantes, el tiempo de creación del archivo de calendario es mayor. Esto es debido a que, mientras más participantes, más partidos habrán y más datos tendrán que ser escritos en dichos archivos. En cuanto al cambio en tiempo del torneo, no se notan diferencias ni patrones notables en esta área.

En la tabla 7.4, y 7.5 para los casos que dan UNSAT, se evidencia la relación que tiene cada tipo de torneo con el número de conflictos que se generan al momento de tratar de encontrar una solución. En cuanto a la cantidad de participantes, mientras más tenga, más conflictos se generan ya que tendrá que tener en cuenta más partidos al momento de asignarle a uno un espacio en el calendario. En cuanto al tiempo por torneo, mientras menos tiempo tenga un torneo, más conflictos se generan debido a que el torneo tendrá menos espacio de holgura para colocar partidos. Algo bastante parecido sucede en las tablas 7.6 y 7.7, pero para estas tablas tomamos en cuenta el número de decisiones que se tomaron durante el proceso de encontrar la solución de los casos de prueba. Estas decisiones representan el número de variables escogidas por el algoritmo, y no tomadas por Unit Propagation, durante su ejecución.

En las tablas 7.8 y 7.9 notamos el tiempo promedio que toma el proceso en encontrar una solución, o en su defecto encontrar que no tiene solución, para los casos de prueba tomados en cuenta. Mientras más participantes, el programa tarda más en encontrar un SAT o UNSAT, debido al incremento en cálculos a causa de los demás factores explicados anteriormente, como el número de conflictos y decisiones.

Para la tabla 7.10, vemos el tiempo que tomó calcular cada restricción para los 70 casos, divididos por su número de participantes. La restricción que más costó calcular para todos los tipos de casos de prueba fue la restricción que dice que dos juegos no pueden ocurrir al mismo tiempo, y esto se veía venir ya que esta es la restricción que representa el verdadero problema a solventar, mientras que el resto pueden verse como factores que aumentan el nivel de dificultad de encontrar su solución. Se agregó una columna que refleja el tiempo promedio que toma el programa durante los cálculos previos a calcular todas las restricciones, en donde se obtiene la lista de partidos a jugar y algunos coeficientes necesarios para los cómputos del proceso. Podemos notar que para todos los casos este tiempo es despreciable, o casi 0. Por último, al igual que con el resto de los cálculos, mientras más participantes tenga el torneo, más tiempo se tardará el programa por restricción.