Universidad Politécnica de Cataluña

Facultad de Matemáticas y Estadística

Planificación del calendario de una liga deportiva

Práctica III

Autores:

Eduardo Peña

Laia Pomar

Diciembre 2019

1. Introducción

Los problemas de programación lineal entera son aquellos problemas de programación lineal en los que las variables toman valores enteros. En el caso de esta práctica, se nos plantea el problema de organizar el calendario de partidos de una liga con equipos de dos distintas regiones, donde los equipos de regiones diferentes tienen que jugar un cierto número de partidos interdivisionales entre ellos y un cierto número de partidos intradivisionales con equipos de la otra región.

Para formular este problema se nos da el número de equipos n, que es natural, par y mayor o igual a 4. La mitad de los equipos, $n/2$, competirá en una región y la mitad restante en la otra región. El objetivo del problema es que al final de la temporada se hayan jugado r partidos intradivisionales entre cada pareja de equipos de la misma división, y s partidos interdivisionales entre cada pareja de equipos de divisiones distintas.

Tal y como indica el enunciado del problema se priorizará que los partidos interdivisionales se jueguen antes en el tiempo que los interdivisionales.

2. Formulación matemática del problema

A continuación se ha Formulado es problema como un problema de programación lineal entera (PLE). Como ya se ha dicho, disponemos de los parámetros n, r, s que nos definen el número total de equipos, la cantidad de partidos intradivisionales y la de partidos interdivisionales que se quieren jugar, respectivamente.

Hemos definidos los siguientes conjuntos que usaremos más adelante en nuestra formulación del problema. Estos conjuntos son: el conjunto $PrimDiv = \{1,...,\frac{n}{2}\}$ y el $SegDiv = \{\frac{n}{2}+1,...,n\}$ que contienen a los equipos de la primera y de la segunda división respectivamente; el conjunto $Todos = PrimDiv \cup SegDiv$; y el conjunto $Jorn = \{1,...,K\}$ que contiene todas las jornadas de la liga. Como se verá más adelante en el apartado **3. Factibilidad y jornadas mínimas**, el elemento K del conjunto $Jorn$ será iugal a $max\{r(n/2-1)+sn/2, rn/2\}$, de manera que este problema no sufra casos de infactibilidad.

Las variables para este problema las hemos planteado para que su valor nos indique qué partidos se juegan en cada jornada, de manera que asociaremos variables binarias x_{ijk} a la jornada k y a la pareja de equipos (i, j), de forma que si su valor es 1 significa que el partido entre los dos equipos se realiza en la jornada indicada, y si es 0, es que no se juega en dicha jornada. Respecto a la función objetivo, queremos maximizar un vector de costes que nos asegure que el calendario resultante priorice que los partidos intradivisionales se jueguen tan tarde como sea posible. Así que, como nos indica el enunciado de este problema, asociaremos los siguientes coeficientes de costes a cada variable:

$$c_{ijk} = \begin{cases} 0 & \text{si } (i,j) \text{ es un partido interdivisional} \\ 0 & \text{si } (i,j) \text{ es un partido intradivisional y } k = 1 \\ 2^{k-2} & \text{si } (i,j) \text{ es un partido intradivisional y } k \geq 2 \end{cases}$$

De esta manera maximizaremos la función objetivo que la hemos definido de la siguiente manera;

$$\text{máx} \sum_{k \in Jorn \setminus \{1\}} 2^{k-2} \left(\sum_{\substack{i,j \in PrimDiv \\ i<j}} x_{ijk} + \sum_{\substack{i,j \in SegDiv \\ i<j}} x_{ijk} \right)$$

Para asegurarnos que el calendario resultante cumplirá todas las condiciones que se mencionan en el enunciado del problema, hemos impuesto que la función objetivo debe satisfacer las siguientes restricciones:

Primer bloque de restricciones:

$$\sum_{k \in Jorn} x_{ijk} = r, \quad \forall i, j \in PrimDiv \mid i < j \tag{1}$$

$$\sum_{k \in Jorn} x_{ijk} = r, \quad \forall i, j \in SegDiv \mid i < j \tag{2}$$

$$\sum_{k \in Jorn} x_{ijk} = s, , \quad \forall i \in PrimDiv, \forall j \in SegDiv \tag{3}$$

Segundo bloque de restricciones:

$$\sum_{\substack{i \in Todos \\ i<l}} x_{ilk} + \sum_{\substack{j \in Todos \\ j>l}} x_{ljk} \leq 1, \quad \forall k \in Jornada, \forall l \in Todos \tag{4}$$

Restricción (1): Imponemos que la suma de los partidos entre una pareja de equipos de la primera división sea r. Es decir, que se jueguen todos los partidos intradivisionales en la primera división.

Restricción (2): Imponemos que la suma de los partidos entre una pareja de equipos de la segunda división sea r. Es decir, que se jueguen todos los partidos intradivisionales en la segunda división.

Restricción (3): Imponemos que los partidos entre cada pareja de equipos de divisiones distintas sea igual a s. Es decir, que se jueguen todos los partidos interdivisionales.

Restricción (4): Aquí imponemos que, en cada jornada, cada equipo juegue como máximo un partido.

Finalmente y juntándolo todo, la formulación del problema nos quedaría de la siguiente manera:

$$(PLE) \begin{cases} \text{máx} & \displaystyle\sum_{k \in Jorn \setminus \{1\}} 2^{k-2} \left(\sum_{\substack{i,j \in PrimDiv \\ i<j}} x_{ijk} + \sum_{\substack{i,j \in SegDiv \\ i<j}} x_{ijk} \right) & \\ \text{s.a:} & \displaystyle\sum_{k \in Jorn} x_{ijk} = r, & \forall i, j \in PrimDiv \mid i < j \\ & \displaystyle\sum_{k \in Jorn} x_{ijk} = r, & \forall i, j \in SegDiv \mid i < j \\ & \displaystyle\sum_{k \in Jorn} x_{ijk} = s, & \forall i \in PrimDiv, \forall j \in SegDiv \\ & \displaystyle\sum_{\substack{i \in Todos \\ i<l}} x_{ilk} + \sum_{\substack{j \in Todos \\ j>l}} x_{ljk} \leq 1, & \forall k \in Jornada, \forall l \in Todos \end{cases}$$

3. Factibilidad y jornadas mínimas

A la hora de resolver este problema nos encontramos que dados ciertos n, r, s el problema es infactible. Esto es debido a que hemos supuesto que en nuestro modelo no descansará ningún equipo ningún día, y por tanto, que se jugarían todos los partido en la mínima cantidad de jornadas posibles, es decir, $r(n/2-1)+sn/2$ jornadas. Sin embargo, en algunos casos no existe ninguna combinación de posibles partidos que cumplan las restricciones de nuestro modelo, que impone que no haya descansos. Para resolver este problema hemos observado lo siguiente:

El objetivo es calcular el número mínimo de jornadas necesarias para que los n equipos puedan jugar r partidos intradivisionales i s partidos interdivisionales. Distinguiremos entre el caso en el que $n/2$ sea par y en el que sea impar.

Cuando $n/2$ es par, podemos considerar el caso $r = 0$ y $s = 1$. En este caso podemos conseguir, sin que ningún equipo descanse, que cada equipo juegue exactamente una vez contra cada uno de los equipos de la división contraria. Para probar esto podemos proceder de la siguiente manera: numeramos los equipos de cada liga del 0 al $n/2$ - 1. Entonces en la jornada k, competirán el equipo y de una liga contra el equipo $i + n/2 \pmod{n/2}$ de la liga contraria. De esta manera, después de $n/2$ jornadas se habrán disputado todos los partidos. Para el caso $r = 1$, $s = 0$, es fácil ver que se pueden jugar todos los partidos en $n/2$ - 1 jornadas.

Por tanto, generalizando los casos anteriores, es posible combinar grupos de $n/2$ o $n/2$ - 1 jornadas como los mencionados con anterioridad de tal manera que se jueguen los partidos en $r(n/2-1)+sn/2$ jornadas, que es el mínimo posible como indica el enunciado de la práctica. En conclusión, dada una n tal que $n/2$ sea par, siempre podremos encontrar una solución factible imponiendo que no haya descansos.

Cuando $n/2$ es impar, podemos ver de nuevo que con $r = 0$, $s = 1$, podemos distribuir los partidos en $n/2$ jornadas de la misma manera que lo hacíamos para el caso par. Así que los partidos interdivisionales sobrantes no suponen un problema.

En cambio, a diferencia del caso par, en el caso impar los problemas aparecen para los partidos intradivisionales. En el caso $r = 1$, $s = 0$ no es posible distribuir los partidos en $n/2-1$ jornadas, ya que, al haber un número impar de equipos y que los enfrentamientos son 2 a 2, habrá 1 equipo que se verá forzado a descansar, pero podemos conseguir que se jueguen todos los partidos en $n/2$ jornadas como mínimo. Esto es debido a que como cada liga tiene un equipo que descansa, estos equipos pueden jugar entre si un partido interdivisional. Observamos que, para que no haya descansos por cada uno de los r partidos que tiene que jugar cada equipo, se tienen que jugar $n/2$ partidos interdivisionales. Debido a esto, dado un s cualquiera, si $r = sn/2$, entonces es posible jugar sin descansar, ya que en total se juegan $s(n/2)^2$ partidos interdivisionales y los podemos organizar para que no se repitan. Entonces, en el caso $r \leq sn/2$, el problema será factible por el mínimo número de jornadas $r(n/2-1)+sn/2$.

No obstante, cuando $r > sn/2$ no es factible para el mínimo número de jornadas, ya que la cantidad de partidos interdivisionales no son suficientes como para llenar los descansos de los partidos intradivisionales. En este caso, simplemente nos centramos en hacer r veces el conjunto de $n/2$ partidos intradivisionales y con los equipos que descansan hacemos los partidos interdivisionales hasta acabarlos. De esta manera vemos que la cantidad de partidos que se jugarán es como mínimo $rn/2$.

En conclusión, para el caso $n/2$ impar, la suma mínima de partidos necesarios para jugar todos los partidos dados r y s será $max\{r(n/2-1)+sn/2, rn/2\}$, e indicando esto en el código AMPL de la sección siguiente, solucionamos los posibles problemas de infactibilidad del problema, y podremos encontrar un calendario (nótese que la solución no es única) óptimo para cualquier tripleta (n, r, s).

4. Modelos AMPL

Para encontrar una solución a este problema ha sido necesaria la utilización y la implementación de AMPL, y por esa razón, se ha trasladado la formulación de este problema a este lenguaje de programación, culminando en el resultado siguiente:

```
#Parametros
param n;
param s;
param r;
param  K = max((r*(n/2 -1) + s*n/2),r*n/2);

set todos:= 1..n;
set prim_div:= 1..(n/2);
set seg_div:= (n/2)+1..n;
set jornada:= 1..K;

var x{i in todos, j in todos, k in jornada: i != n and j > i}, binary;
var desc{i in todos}, binary;

#Funcion objetivo
maximize c:
sum{k in jornada: k != 1}2^(k-2)*(sum{i in prim_div, j in prim_div: i != n/2 and j > i} x[i,j,k] + sum{i in seg_div, j
      in seg_div: i != n and j > i} x[i,j,k]);

#Primer bloque restricciones
subject to primera_division_intra{i in prim_div, j in prim_div: i != n/2 and j > i}:
  sum{k in jornada} x[i,j,k] = r;
subject to segunda_division_intra{i in seg_div, j in seg_div: i != n and j > i}:
  sum{k in jornada} x[i,j,k] = r;
subject to partidos_inter{i in prim_div, j in seg_div}:
  sum{k in jornada} x[i,j,k] = s;

#Segundo bloque restricciones
subject to un_partido_por_jornada{l in todos, k in jornada}:
   (sum{i in todos: i < l} x[i,l,k] + sum{j in todos: j > l} x[l,j,k]) <= 1;
```

Además de modelizar el problema en AMPL, también se ha creado el siguiente archivo .run para ejecutar el programa y mostrar la solución de manera inteligible y agradable, tal y como se puede apreciar en el apartado 5, donde se muestran algunos ejemplos de resoluciones del problema. Archivo .run:

```
reset;
option solver cplex;
model calendario.mod;
data datos.dat;
solve;
for{k in jornada}{
  printf "Jornada %d de %d\n", k, K;
  for {i in todos, j in todos: j > i and i != n}{
    if x[i,j,k] = 1 then {
      printf "    Equipo %d - Equipo %d \n",i,j;
    }
  }
  printf "\n";
}
```

5. Algunos ejemplos resueltos

A continuación se muestran algunos ejemplos de calendarios óptimos juntamente con el valor de la función objetivo para algunas ternas (n, r, s). Nótese que el primer ejemplo cumple que $n/2$ es par y por tanto se jugarán todos los partidos en un mínimo de $r(n/2-1)+sn/2$ jornadas, mientras que en el segundo y tercer ejemplo, $n/2$ es impar, y por tanto los partidos serán jugados en un mínimo de $max\{r(n/2-1)+sn/2, rn/2\}$. En el segundo ejemplo será en $rn/2$ jornadas ya que $r > sn/2$, mientras que en el tercero será en $r(n/2-1)+sn/2$, ya que $r < sn/2$.

Ejemplo de .dat 1 -

```
param n := 8;
param r := 3;
param s := 2;
```

Ejemplo de .dat 2 -

```
param n := 6;
param r := 7;
param s := 2;
```

Ejemplo de .dat 3 -

```
param n := 6;
param r := 2;
param s := 3;
```

Resolución .dat 1. $n = 8$ $r = 3$ $s = 2$

CPLEX 12.9.0.0: optimal integer solution; objective 261632
247 MIP simplex iterations
0 branch-and-bound nodes

Jornada 1 de 17
Equipo 1 - Equipo 5
Equipo 2 - Equipo 6
Equipo 3 - Equipo 8
Equipo 4 - Equipo 7

Jornada 2 de 17
Equipo 1 - Equipo 5
Equipo 2 - Equipo 6
Equipo 3 - Equipo 8
Equipo 4 - Equipo 7

Jornada 3 de 17
Equipo 1 - Equipo 7
Equipo 2 - Equipo 5
Equipo 3 - Equipo 6
Equipo 4 - Equipo 8

Jornada 4 de 17
Equipo 1 - Equipo 7
Equipo 2 - Equipo 5
Equipo 3 - Equipo 6
Equipo 4 - Equipo 8

Jornada 5 de 17
Equipo 1 - Equipo 8
Equipo 2 - Equipo 7
Equipo 3 - Equipo 5
Equipo 4 - Equipo 6

Jornada 6 de 17
Equipo 1 - Equipo 8
Equipo 2 - Equipo 7
Equipo 3 - Equipo 5
Equipo 4 - Equipo 6

Jornada 7 de 17
Equipo 1 - Equipo 6
Equipo 2 - Equipo 8
Equipo 3 - Equipo 7
Equipo 4 - Equipo 5

Jornada 8 de 17
Equipo 1 - Equipo 6
Equipo 2 - Equipo 8
Equipo 3 - Equipo 7
Equipo 4 - Equipo 5

Jornada 9 de 17
Equipo 1 - Equipo 2
Equipo 3 - Equipo 4
Equipo 5 - Equipo 7
Equipo 6 - Equipo 8

Jornada 10 de 17
Equipo 1 - Equipo 3
Equipo 2 - Equipo 4
Equipo 5 - Equipo 8
Equipo 6 - Equipo 7

Jornada 11 de 17
Equipo 1 - Equipo 2
Equipo 3 - Equipo 4
Equipo 5 - Equipo 8
Equipo 6 - Equipo 7

Jornada 12 de 17
Equipo 1 - Equipo 4
Equipo 2 - Equipo 3
Equipo 5 - Equipo 6
Equipo 7 - Equipo 8

Jornada 13 de 17
Equipo 1 - Equipo 4
Equipo 2 - Equipo 3
Equipo 5 - Equipo 8
Equipo 6 - Equipo 7

Jornada 14 de 17
Equipo 1 - Equipo 2
Equipo 3 - Equipo 4
Equipo 5 - Equipo 7
Equipo 6 - Equipo 8

Jornada 15 de 17
Equipo 1 - Equipo 3
Equipo 2 - Equipo 4
Equipo 5 - Equipo 6
Equipo 7 - Equipo 8

Jornada 16 de 17
Equipo 1 - Equipo 4
Equipo 2 - Equipo 3
Equipo 5 - Equipo 7
Equipo 6 - Equipo 8

Jornada 17 de 17
Equipo 1 - Equipo 3
Equipo 2 - Equipo 4
Equipo 5 - Equipo 6
Equipo 7 - Equipo 8

Resolución .dat 2. $n = 6$ $r = 7$ $s = 2$

CPLEX 12.9.0.0: optimal integer solution within mipgap or absmipgap; objective 2097150
411 MIP simplex iterations
0 branch-and-bound nodes

Jornada 1 de 21
Equipo 1 - Equipo 3
Equipo 2 - Equipo 5
Equipo 4 - Equipo 6

Jornada 2 de 21
Equipo 1 - Equipo 2
Equipo 3 - Equipo 5
Equipo 4 - Equipo 6

Jornada 3 de 21
Equipo 1 - Equipo 3
Equipo 4 - Equipo 6

Jornada 4 de 21
Equipo 1 - Equipo 2
Equipo 4 - Equipo 5

Jornada 5 de 21
Equipo 1 - Equipo 2
Equipo 3 - Equipo 4
Equipo 5 - Equipo 6

Jornada 6 de 21
Equipo 1 - Equipo 2
Equipo 3 - Equipo 6
Equipo 4 - Equipo 5

Jornada 7 de 21
Equipo 1 - Equipo 5
Equipo 2 - Equipo 3
Equipo 4 - Equipo 6

Jornada 8 de 21
Equipo 1 - Equipo 2
Equipo 3 - Equipo 6
Equipo 4 - Equipo 5

Jornada 9 de 21
Equipo 1 - Equipo 3
Equipo 2 - Equipo 4
Equipo 5 - Equipo 6

Jornada 10 de 21
Equipo 1 - Equipo 6
Equipo 2 - Equipo 3
Equipo 4 - Equipo 5

Jornada 11 de 21
Equipo 1 - Equipo 5
Equipo 2 - Equipo 3
Equipo 4 - Equipo 6

Jornada 12 de 21
Equipo 1 - Equipo 2
Equipo 3 - Equipo 4
Equipo 5 - Equipo 6

Jornada 13 de 21
Equipo 1 - Equipo 3
Equipo 2 - Equipo 6
Equipo 4 - Equipo 5

Jornada 14 de 21
Equipo 2 - Equipo 3
Equipo 5 - Equipo 6

Jornada 15 de 21
Equipo 1 - Equipo 3
Equipo 2 - Equipo 6
Equipo 4 - Equipo 5

Jornada 16 de 21
Equipo 1 - Equipo 2
Equipo 3 - Equipo 5
Equipo 4 - Equipo 6

Jornada 17 de 21
Equipo 1 - Equipo 4
Equipo 2 - Equipo 3
Equipo 5 - Equipo 6

Jornada 18 de 21
Equipo 1 - Equipo 3
Equipo 2 - Equipo 4
Equipo 5 - Equipo 6

Jornada 19 de 21
Equipo 1 - Equipo 4
Equipo 2 - Equipo 3
Equipo 5 - Equipo 6

Jornada 20 de 21
Equipo 1 - Equipo 6
Equipo 2 - Equipo 3
Equipo 4 - Equipo 5

Jornada 21 de 21
Equipo 1 - Equipo 3
Equipo 2 - Equipo 5
Equipo 4 - Equipo 6

Resolución .dat 3. $n = 6$ $r = 2$ $s = 3$

CPLEX 12.9.0.0: optimal integer solution; objective 8064
184 MIP simplex iterations
0 branch-and-bound nodes

Jornada 1 de 13
Equipo 1 - Equipo 6
Equipo 2 - Equipo 4
Equipo 3 - Equipo 5

Jornada 2 de 13
Equipo 1 - Equipo 6
Equipo 2 - Equipo 4
Equipo 3 - Equipo 5

Jornada 3 de 13
Equipo 1 - Equipo 5
Equipo 2 - Equipo 6
Equipo 3 - Equipo 4

Jornada 4 de 13
Equipo 1 - Equipo 5
Equipo 2 - Equipo 6
Equipo 3 - Equipo 4

Jornada 5 de 13
Equipo 1 - Equipo 4
Equipo 2 - Equipo 5
Equipo 3 - Equipo 6

Jornada 6 de 13
Equipo 1 - Equipo 6
Equipo 2 - Equipo 5
Equipo 3 - Equipo 4

Jornada 7 de 13
Equipo 1 - Equipo 4
Equipo 2 - Equipo 5
Equipo 3 - Equipo 6

Jornada 8 de 13
Equipo 1 - Equipo 2
Equipo 3 - Equipo 6
Equipo 4 - Equipo 5

Jornada 9 de 13
Equipo 1 - Equipo 3
Equipo 2 - Equipo 4
Equipo 5 - Equipo 6

Jornada 10 de 13
Equipo 1 - Equipo 3
Equipo 2 - Equipo 6
Equipo 4 - Equipo 5

Jornada 11 de 13
Equipo 1 - Equipo 5
Equipo 2 - Equipo 3
Equipo 4 - Equipo 6

Jornada 12 de 13
Equipo 1 - Equipo 2
Equipo 3 - Equipo 5
Equipo 4 - Equipo 6

Jornada 13 de 13
Equipo 1 - Equipo 4
Equipo 2 - Equipo 3
Equipo 5 - Equipo 6

Como podemos observar, en todos los ejemplos se cumple lo predicho en el apartado **3. Factibilidad y jornadas mínimas**. En el primer ejemplo, al ser $n/2$ par, la mínima cantidad mínima de jornadas es $r(n/2-1)+sn/2 = 3(4-1)+2\times 4 = 17$. En el segundo ejemplo, al ser $n/2$ impar y $r > sn/2$ la cantidad mínima de jornadas es $rn/2 = 7\times 3 = 21$. Finalmente, en el último ejemplo, $n/2$ vuelve a ser impar, pero $r \leq sn/2$, por tanto el mínimo de jornadas es $r(n/2-1)+sn/2 = 2(3-1)+3\times 3 = 13$.

Como hemos comprobado, se cumple lo dicho y el problema con nuestra formulación es factible por cualquier tripleta (n, r, s) que cumpla las hipótesis del enunciado.