UNIVERSIDAD DE BUENOS AIRES
Facultad de Ciencias Exactas y Naturales
Departamento de Matemática

Tesis de Licenciatura

Programar torneos de fútbol equiparando descansos: el caso de la Liga Profesional Argentina

Diego Nicolás Marucho

Director: Guillermo Durán

14/11/2022

Agradecimientos

Hay mucha gente a la que me gustaría agradecerle todo lo que hizo por mí en este increíble camino de casi 7 años. Todos, desde su lugar y de distintas formas, fueron muy importantes para motivarme, ayudarme o escucharme a lo largo de mi carrera.

Primero que nada y antes que todos le agradezco profundamente a mi familia, quienes tuvieron que soportarme y aguantarme desde el primer día. Les tocó convivir conmigo, mis "no puedo, tengo que estudiarçonstante y mis mal humores por no poder estar al día con las clases. Siempre me escucharon sin entender absolutamente nada de lo que les contaba y así todo me sonreían como si nada. Mención especial para mi mamá que hizo todo para que pueda enfocarme en el estudio durante todos estos años. Me apoyó y ayudó desde el día que tuve que elegir una carrera, acompañándome a exactas para esa primera charla sobre la carrera de Licenciatura en ciencias matemáticas cuando todavía estaba en el colegio. Gracias por compartirme tu amor hacia esta hermosa facultad (bióloga de exactas) que me hizo tan feliz.

Gracias a Sol, mi pareja, a quien conocí en el CBC y llevamos casi 6 años juntos. Gracias por escucharme siempre y entender más que nadie lo que significaba la facultad para mi. Nunca tuvo ningún problema si tenía que estudiar y nos teníamos que ver poco o incluso no podíamos vernos por un fin de semana. Festejó todos y cada uno de mis exámenes conmigo como si hubiesen sido suyos.

A mis amigos y amigas del CBC, el mejor grupo que pude haber conocido para transitar ese año. Fue todo risas, charlas en el río y largos días copando la biblioteca del pabellón 2 para estudiar antes de los parciales.

A Lei, Lu, Juancito, Leti, Fran y Agus, que junto con Sol, son lo mejor que deja esta facultad. Gracias por los viajes, las distracciones y los chistes eternos.

A los infaltables amigos y amigas de matemática, Cande, Cris, Fer y Fede, que me ayudaron tanto y me tuvieron tanta paciencia estudiando.

A Fede, mi fiel compañero en todas las materias de matemática aplicada, con quien tuvimos que soportar una larga pandemia y la sacamos adelante.

A mi grupo de amigos del colegio, que a pesar de los constantes "no puedo, tengo que estudiar", "mañana me levanto temprano a estudiar", etc. , siempre intentaron sacarme un rato del estudio para hablar de fútbol.

A Willy, que fue quien el primero en presentarme una materia en la que dije "para esto me anote en esta carrera". Mostrarme como mezclar dos cosas tan hermosas como la matemática y el deporte fue una de las cosas más valiosas que me enseñaron. Gracias por finalizar cada conversación con un "Y que gane San Lorenzo".

A Joaco, quien en muy poco tiempo hizo muchísimo por mi. Desde referirme en su trabajo hasta enseñarme cada día lo lindo que es optimizar. Siempre se hizo un rato para darme recomendaciones, ayudarme a encontrar errores o contarme sobre su tesis.

Índice general

Índice de figuras

Índice de cuadros

Introducción

La *matemática aplicada* puede definirse como la utilización de métodos o herramientas matemáticas para resolver problemas de la vida real. Muchos de estos métodos resultaron muy efectivos también para el estudio de otras ramas de la ciencia como lo son física, química, biología, economía, finanzas, entre otras. A diferencia de la matemática conocida como "pura"o matemática académica, en la cual el desarrollo se da en la misma matemática, en el mundo aplicado se busca desarrollar otras áreas. Algunas áreas de la matemática donde más se utilizan las matemáticas aplicadas para resolver problemas reales son: Álgebra lineal, Cálculo diferencial, Estadística y Optimización. Esta última es la que nos compete en este trabajo.

En términos generales, optimizar hace referencia a tener la capacidad de resolver algo de la forma más eficiente posible. La *optimización* o programación matemática puede describirse simplemente como encontrar el *mejor* punto de entre un conjunto de puntos que satisfacen ciertas restricciones. En general, existe una función (llamada función objetivo) la cual será minimizada o maximizada, dependiendo de la naturaleza del problema. Existen muchos tipos de problemas que pueden resolverse optimizando. A su vez, existen diversos algoritmos o métodos que sirven de ayuda para resolver cada uno de estos tipos de problemas. En este trabajo nos enfocaremos en el campo de la *Programación Lineal*.

Un problema de programación lineal se caracteriza por tener tanto una función objetivo como todas sus restricciones lineales en las variables. Estas últimas en general se presentan mediante un sistema de ecuaciones o inecuaciones lineales. Este tipo de problemas representan un campo muy importante dentro de la optimización, ya que muchos problemas prácticos pueden plantearse de manera lineal. Algunos casos especiales como los problemas de flujo crecieron tanto en los últimos años que los investigadores se concentraron en generar algoritmos especiales para su resolución. En cuanto a la resolución de este tipo de problemas, el algoritmo más utilizado es el algoritmo *simplex*.

En algunos casos, se tienen problemas en los que las variables son enteras (y no reales). Estos pertenecen al conjunto de problemas de la *Programación entera*. Otra posibilidad, que es la que nos compete en el trabajo, es tener variables de ambos tipos. Estos problemas se conocen como problemas de *Programación mixta*. El problema con variables reales es más fácil de resolver, por lo que una técnica usual para resolver el problema mixto es relajarlo y considerar las variables enteras como reales. La solución de dicho problema relajado sirve como cota para el problema original. El método más famoso para resolver este tipo de problemas es *Branch and*

Bound.

Sports scheduling

Una de las tantas aplicaciones que tienen los problemas de programación mixta se da en el mundo del deporte. Hoy en día, el deporte despierta el interés de una enorme multitud de personas a lo largo del mundo y tiene un fuerte impacto en la economía global, sobretodo en algunas disciplinas como el fútbol. Como no podía ser de otra forma, la matemática comenzó a hacerse presente en este campo. Dentro de todos los problemas posibles en los que la matemática puede aportar soluciones al deporte, se encuentra el armado de Fixtures. Así es como surgió el concepto de *Sports scheduling*. La mayoría de los problemas de este tipo constan de determinar la fecha y el lugar donde se va a jugar cada partido de un torneo.

Si bien armar un fixture de fútbol suena accesible en cuanto a su dificultad, es un problema que se vuelve muy difícil con rapidez. El sólo hecho de aumentar el número de equipos participantes lo hace un problema cada vez más complejo. Sumado a esto, existen muchas restricciones que podemos separar en dos grupos: internas y externas. Las restricciones internas son todas aquellas que necesitamos para la creación del fixture en sí. Algunos ejemplos de estas son que cada equipo juegue una única vez por fecha, que cada equipo juegue una vez o dos contra el resto de los equipos, entre otras. Por otro lado, tenemos las externas, que en general son impuestas por los organizadores. Si bien estas restricciones son evitables, muchas veces son de extrema importancia. En este grupo podemos encontrar pedidos como que los equipos no jueguen muchos partidos consecutivos de local o de visitante, restricciones televisivas o incluso de seguridad (que no sean locales dos equipos cuyas canchas están muy cerca).

Existen muchos criterios que son considerados a la hora de generar el mejor fixture para un campeonato. Los mismos dependen también del formato del torneo. Se dice que un torneo es del tipo *Single Round Robin* (o simplemente Round Robin) cuando todos los equipos juegan entre sí una única vez a lo largo del torneo, mientras que un torneo *Doble Round Robin* todos los equipos deben enfrentarse dos veces en todo el torneo. En cuanto a los criterios para crear un fixture, algunos muy conocidos y estudiados en el último tiempo son: minimizar la distancia total viajada por los equipos, minimizar el efecto de arrastre que sufren los equipos y minimizar la cantidad total de *breaks* en un torneo. Decimos que existe un *break* cuando un equipo debe jugar dos partidos consecutivos de local o de visitante.

En este trabajo se muestra una aplicación a un criterio que todavía no fue tan estudiado en el mundo del *Sports scheduling*. El problema consiste en asignar los días de un fixture minimizando la diferencia que existen en los descansos de los equipos. Esto es, que si dos equipos deben enfrentarse en cierta fecha, que la fecha anterior hayan jugado el mismo período o lo más cercano posible. En este problema, el fixture está dado, es decir, ya se sabe en cada fecha cuales serán los cruces de los equipos. Se denotará a un *período* como la separación temporal de cada fecha, el cual puede ser un horario, un día, etc. En dichos *períodos* se jugará una cierta

cantidad de partidos que puede ser fija o no a lo largo del campeonato. A este problema se lo conoce como *The Rest Difference Problem* y el objetivo será aplicarlo a la Liga profesional de Fútbol (la competición de la primera división del fútbol argentino). Una variante del trabajo será generar también el fixture inicial tomado como input, el cual puede crearse mediante algún modelo de programación mixta, independientemente del segundo modelo, bajo alguno de los criterios dichos arriba, como por ejemplo, minimizar los *breaks*. La ventaja de tomar el fixture dado por los organizadores, es que ya cumple con ciertas restricciones que ellos desean y creen importantes.

Contenido

La distribución del presente trabajo será la siguiente:

En el Capítulo 1 daremos los conceptos básicos necesarios de la optimización. Dentro de este campo, nos concentraremos en la programación lineal, introduciendo el método más famoso para resolver este tipo de problemas: *el método simplex*.

Por último, hablaremos de los problemas enteros y mixtos y de un algoritmo importante para resolverlos: *Branch and Bound*.

En el Capítulo 2 presentaremos el problema de *The Difference Problem* comenzando con el trabajo previo abordado en el tema y pasando por una variante del problema que se conoce como *The Rest Mismatch Problem*. Aquí se verán también algunos métodos y heurísticas existentes para resolver el problema.

El Capítulo 3 explica brevemente el formato de juego de la Liga Profesional de Fútbol seguido del modelo de programación mixta que se utilizará.

En el Capítulo 4 se darán a conocer los resultados obtenidos del modelo para distintas corridas, modificando las restricciones del problema de acuerdo a distintas situaciones.

En el Capítulo 5 se concluirá el trabajo analizando los resultados obtenidos y proponiendo opciones de trabajo futuro relacionadas al problema.

Capítulo 1

Optimización matemática

La optimización matemática (también conocida como programación matemática) es una rama de las matemáticas aplicadas enfocada en encontrar al mejor "elemento", sujeto a una serie de criterios (llamadas restricciones), entre un conjunto de opciones disponibles. Si bien la optimización tiene una fuerte base matemática, su aplicación abarca muchos aspectos de la vida real, incluso algunos muy cotidianos.

En su forma más simple, un problema de optimización consta de los siguientes componentes:

- Variables de decisión: $x_1, ..., x_n$
- Restricciones: Se debe satisfacer que $(x_1, ..., x_n) \in S \subseteq \mathbb{R}^n$. El conjunto S se denomina conjunto de soluciones factibles o región de factibilidad.
- Función objetivo: $f : \mathbb{R}^n \mapsto \mathbb{R}$

En un problema de optimización se busca minimizar o maximizar la función objetivo sujeto a las restricciones. Es decir, al minimizar (maximizar), buscamos un elemento $x_0 \in S$ tal que $f(x_0) \leq f(x) \quad \forall x \in S \quad (\geq$ al maximizar). En este caso diremos que en x_0 se alcanza el óptimo con valor $f(x_0)$. Puesto que lo siguiente es equivalente:

$$f(x_0) \geq f(x) \Leftrightarrow -f(x_0) \leq -f(x),$$

es suficiente con concentrarse en solo uno de los dos tipos de problemas, ya que podemos fácilmente transformar uno en el otro. Dicho esto, suelen plantearse los problemas minimizando la función objetivo, y es lo que utilizaré de aquí en adelante.

Se define un mínimo local x^* como un elemento para el cual existe un $\delta > 0$ tal que $\forall x \in S$ donde $\|x - x^*\| \leq \delta$, se tiene que $f(x^*) \leq f(x)$. Esto es, que en alguna región alrededor de x^* todos los valores de la función objetivo son mayores o iguales a la evaluación de x^*. Generalmente, en un problema de minimización, a menos que la función objetivo sea convexa, pueden existir varios mínimos locales. Por otro lado, un mínimo global es aquel punto para el cual su evaluación en la función objetivo es menor o igual que todos los puntos de S.

En el caso en el que $S = \mathbb{R}^n$ se dice que el problema es irrestricto. En el caso contrario, se representará al conjunto de restricciones S como un conjunto finito de

desigualdades:

$$S = \{g_i \leq 0 : i = 1, ...m\}$$

donde cada $g_i : \mathbb{R}^n \mapsto \mathbb{R}$ representa alguna característica o condición que se desea incorporar. En resumen, un problema estándar de optimización se ve da la siguiente forma:

$$\begin{aligned} & Min\ f(x_1, ..., x_n) \\ Sujeto\ a: & g_1(x_1, ..., x_n) \leq 0 \\ & g_2(x_1, ..., x_n) \leq 0 \\ & \vdots \\ & g_m(x_1, ..., x_n) \leq 0 \end{aligned}$$

Dentro de los problemas de optimización, se encuentran dos grandes familias de problemas: los problemas lineales y los problemas no lineales. Esta caracterización depende de la naturaleza de la función objetivo y las restricciones. En los primeros todas las funciones involucradas son lineales con respecto a las variables. Este tipo de problemas son muy estudiados hoy en día y es el tipo de problema que nos interesa en el presente trabajo.

1.1. Programación lineal

La programación lineal (LP por sus siglas en inglés) es un método que busca alcanzar el mejor resultado de un modelo matemático cuyas restricciones y función objetivo son lineales en las variables.

En general, si $c_1, ..., c_n$ son números reales, f una función que toma valores reales $x = (x_1, ..., x_n)$ definida por:

$$f(x_1, ..., x_n) = c_1x_1 + ... + c_nx_n = \sum_{j=1}^{n} c_jx_j$$

se denomina función lineal. Si f es una función lineal y b un número real, entonces la ecuación:

$$f(x_1, ..., x_n) = b$$

se llama ecuación lineal y las inecuaciones

$$f(x_1, ..., x_n) \leq b$$

$$f(x_1, ..., x_n) \geq b$$

inecuaciones lineales. Tanto las ecuaciones lineales como las inecuaciones lineales serán consideradas como restricciones lineales. Bajo estas definiciones, un problema de programación lineal consta de minimizar (o maximizar) una función lineal sujeto

a finitas restricciones lineales. De esta forma, vamos a representar los problemas LP en su versión estándar de la siguiente manera:

$$\begin{aligned} Min \ & \sum_{j=1}^{n} c_j x_j \\ Sujeto\ a: \ & \sum_{j=1}^{n} a_{ij} x_j = b_i \qquad (i = 1, 2, ..., m) \\ & x_j \geq 0 \qquad (j = 1, 2, ..., n) \end{aligned}$$

donde i representa las diferentes restricciones y j las distintas variables. Observar que es simple transformar cualquier problema lineal en su forma estándar, separando las igualdades en dos desigualdades y utilizando la propiedad dada con el signo menos.

Todos los puntos del espacio que satisfagan todas las restricciones de un problema LP dado se las denomina **solución factible**. Aquella solución factible que minimiza la función objetivo se llama **solución óptima**, y su evaluación en la función es el **valor óptimo** del problema. No todos los problemas LP tienen una única solución, algunos tienen muchas como también existen problemas sin soluciones.

Para introducirnos en el aspecto teórico, es mucho más útil escribir la forma estándar de un LP en su forma matricial:

$$\begin{aligned} Min\ z = c^T x \\ Sujeto\ a: \ Ax = b \\ x \geq 0 \end{aligned}$$

donde $A \in \mathbb{R}^{m \times n}$, $c, x \in \mathbb{R}^n$ y $b \in \mathbb{R}^m$. Se sobreentiende que la última restricción hace referencia a que cada coordenada del vector x es no negativa. Además, vamos a adoptar la notación de x^* para el punto óptimo del problema y z^* el valor óptimo, donde $z^* = c^T x^*$.

Sea K $= \{x \in \mathbb{R}^n \ / \ Ax = b, \ x \geq 0\}$, el conjunto de soluciones factibles del problema. Nos interesa ver que este es un conjunto convexo y para eso veamos que significa esto.

Definición 1. *Un conjunto $S \subseteq \mathbb{R}^n$ se llama **convexo** si para todo $x, y \in S$, el segmento que los une está completamente incluido en S, es decir, $\{\lambda_1 x + \lambda_2 y \ \ | \ \ \lambda_1 + \lambda_2 = 1 \ y \ \lambda_1, \lambda_2 \geq 0\} \subseteq S$.*

Veamos que K es un conjunto convexo.

Teorema 1. *Sea K el conjunto de soluciones factibles del LP y supongamos que $K \neq \emptyset$. Entonces, K es un conjunto convexo.*

Demostración:
Si K contiene un único punto, el resultado es cierto. Supongamos que existen al menos dos soluciones factibles distintas del problema, x^1 y x^2. Tenemos que:

$$Ax^1 = b, x^1 \geq 0$$

$$Ax^2 = b, x^2 \geq 0$$

Sea $x = \lambda x^1 + (1-\lambda)x^2$, con $0 \leq \lambda \leq 1$, una combinación lineal convexa de x^1 y x^2. Es claro que $x \geq 0$. Además, $Ax = A(\lambda x^1 + (1-\lambda)x^2) = \lambda Ax^1 + (1-\lambda)Ax^2 = \lambda b + (1-\lambda)b = b$, por lo que $x \in K$. Luego, K es convexo. □

Este resultado nos da la ventaja de que todas las propiedades de los conjuntos convexos pueden aplicarse al problema lineal. Esto hace que el problema sea más fácil de resolver y nos permite encontrar óptimos globales, algo que no sucede en todos los problemas.

Veamos ahora una serie de definiciones que nos van a servir para introducir el método de simplex.

Definición 2. *Un conjunto formado por la intersección de un número finito de semiespacios afines se llama* ***poliedro****.*

Así, K $= \{x \in \mathbb{R}^n \ / \ Ax = b, \ x \geq 0\}$ es un poliedro.

Definición 3. *La envoltura o* ***cápsula convexa*** *$e(S)$ de un conjunto dado de puntos S es el conjunto de todas las combinaciones lineales convexas de puntos de S.*

$$e(S) := \left\{ \sum_{i \in J} \lambda_i x^i, \quad \forall \{x^i\}_{i \in J} \subseteq S, \quad \sum_{i \in J} \lambda_i = 1, \quad \lambda_i \geq 0, \quad i \in J, \quad J \quad finito \right\}$$

Equivalentemente, la envoltura convexa de un conjunto es el menor conjunto convexo que lo contiene.

Si el conjunto S está formado por un número finito de puntos, la envoltura convexa de S se denomina **polítopo**. En la figura 1.1 podemos ver dos ejemplos de esta definición.

Definición 4. *Sea $S \subseteq \mathbb{R}^n, S \neq \emptyset$, cerrado y convexo. Un punto $x \in S$ se dice punto extremo o* ***vértice*** *de S si no es posible expresar a x como combinación lineal convexa de otros puntos del conjunto S. Esto es, no existen dos puntos distintos $x^1, x^2 \in S$ tal que $x = \lambda x^1 + (1-\lambda)x^2$ para algún λ tal que $0 < \lambda < 1$.*

Si bien los polítopos por definición tienen finitos vértices, esto no quiere decir que estos son acotados. Otra forma de definirlos es a través de un número finito de desigualdades lineales. Y esta definición es la que utilizaremos.

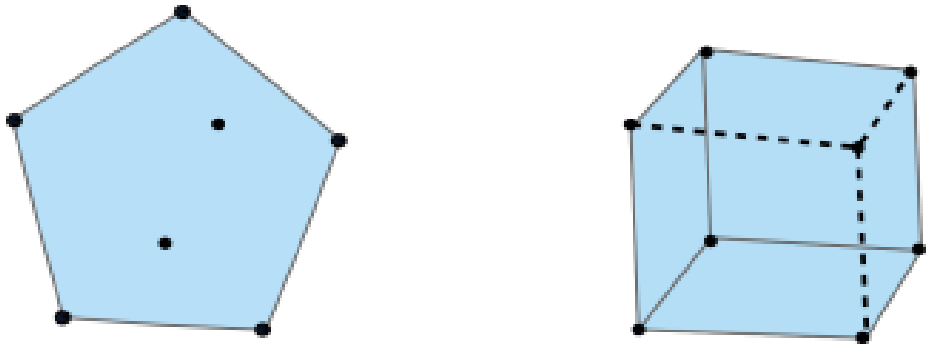


Figura 1.1: Dos polítopos vistos como la cápsula convexa de un conjunto de puntos

Definición 5. *Dados $a \in \mathbb{R}^n$ y $b \in \mathbb{R}$, un* ***hiperplano*** *H en $\mathbb{R}^n$ se define como*

$$H(a,b) = \{x \in \mathbb{R}^n \quad | \quad a.x = b\},$$

donde el vector a se llama vector normal al hiperplano H. Un ***semiespacio*** *en $\mathbb{R}^n$ es un conjunto de la forma*

$$H(a,b)_+ = \{x \in \mathbb{R}^n \quad | \quad a.x \leq b\},$$

con $a \in \mathbb{R}^n - \{0\}$ y $b \in \mathbb{R}^n$.

Notar que también podemos definir a $H(a,b)_-$ análogamente reemplazando el signo $\leq$ por $\geq$. Otra definición útil e importante para esta sección es la de **hiperplano de soporte**.

Definición 6. *Un* ***hiperplano de soporte*** *para un polítopo P es un hiperplano $H(a,b) = \{x \in \mathbb{R}^n \quad | \quad a.x = b\}$ tal que $P \cap H(a,b) \neq \emptyset$ y, además $P \subseteq H(a,b)_+ = \{x \in \mathbb{R}^n \quad | \quad a.x \leq b\}$ o $P \subseteq H(a,b)_- = \{x \in \mathbb{R}^n \quad | \quad a.x \geq b\}$.*

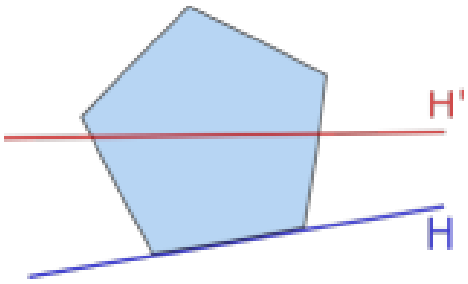


Figura 1.2: H es hiperplano de soporte para el pentágono, mientras que H' no lo es

Definición 7. *Diremos que un hiperplano $H(a,b) \subseteq \mathbb{R}^n$* ***separa*** *dos conjuntos $X, Y \in \mathbb{R}^n$ si $X \subseteq H(a,b)_+$ e $Y_- \subseteq H(a,b)_-$, donde $H(a,b)_+$, $H(a,b)_-$ son los semiespacios determinados por H.*

Teorema 2. *Sea $S \subseteq \mathbb{R}^n$ un conjunto convexo, cerrado y no vacío, y sea $x \in \mathbb{R}^n - S$. Entonces, S y $\{x\}$ pueden separarse por un hiperplano $H(a,b)$ que soporta a S.*

Demostración:
Por ser S cerrado y no vacío, podemos considerar a $\overline{x}$, el punto de S más cercano a x.

Sea $a = \overline{x} - x$ y $b = a.\overline{x}$. Veamos que el hiperplano $H(a,b)$ soporta a S y separa a S de x.

Primero veamos que $a.x < b$. Tenemos que $a.\overline{x} - a.x = a(\overline{x} - x) = a.a$. Puesto que $\overline{x} \neq x$ (ya que uno está en S y el otro no), se tiene que $a \neq 0$ por definición y, por lo tanto, $a.\overline{x} - a.x = a.a > 0$. En consecuencia, $a.x < a.\overline{x} = b$. Luego, todo punto que no está en S, está incluido en $H(a,b)_-$.

Queremos ver que S está incluido en $H(a,b)_+$, es decir, que para todo $y \in S$ se tiene que $a.y \geq b$. Sea $y \in S$ y supongamos, por contradicción, que $a.y < b$. Para cada $\lambda \in (0,1)$ definimos $x^\lambda = (1-\lambda)\overline{x} + \lambda y$. Por ser S convexo y $\overline{x}, y \in S$, se tiene que $x^\lambda \in S \quad \forall \lambda \in (0,1)$. Veamos que para algunos valores suficientemente chicos de λ, se tiene que $d(x, x^\lambda) < d(x, \overline{x})$, lo cual genera una contradicción dado que $\overline{x}$ es el punto de S más cercano a x.

Notemos primero que

$$\begin{aligned} x^\lambda - x &= (1-\lambda)\overline{x} + \lambda y - x \\ &= (\overline{x} - x) + \lambda(y - \overline{x}) \\ &= a + \lambda(y - \overline{x}) \end{aligned}$$

Ahora veamos que la distancia de x^λ a x es menor que la distancia de $\overline{x}$ a x. O lo que es equivalente, ver que $||x^\lambda - x||^2 < ||\overline{x} - x||^2$.

$$\begin{aligned} ||x^\lambda - x||^2 &= (x^\lambda - x)(x^\lambda - x) \\ &= (a + \lambda(y - \overline{x}))(a + \lambda(y - \overline{x})) \\ &= a.a + 2.\lambda a.(y - \overline{x}) + \lambda^2 ||y - \overline{x}||^2 \\ &= ||\overline{x} - x||^2 + \lambda g(\lambda) \end{aligned}$$

donde $g(\lambda) = 2.(a.y - a.\overline{x}) + \lambda ||y - \overline{x}||^2$. Y como habíamos supuesto que $a.y < b = a.\overline{x}$, tenemos que $\lim_{\lambda \to 0} g(\lambda) < 0$ y, por lo tanto, para valores chicos de λ se tiene que $||x^\lambda - x||^2 < ||\overline{x} - x||^2$. Por lo dicho antes, como esto no puede suceder, se tiene que $a.y \leq b \quad \forall y \in S$. Luego, $S \subseteq H(a,b)_+$ y $H(a,b)$ separa a S y $\{x\}$. Puesto que $\overline{x} \in S$, se tiene además que $H(a,b)$ soporta a S. $\square$

Teorema 3. *Sea $S \subseteq \mathbb{R}^n$ un conjunto cerrado no vacío. Si S es convexo, dado $z \in \partial S$, existe un hiperplano $H(a,b)$ soporte de S que contiene a z.*

Demostración:
Sea $z \in \partial S$. Dado $n \in \mathbb{N}$, sabemos que siempre existe un punto z_n en la bola $B(z, \frac{1}{n})$ que no está en S. Más aún, se cumple que $\lim_{n \to \infty} z_n = z$.

Por el teorema 2, para cada uno de estos z_n existe un hiperplano $H(a_n, b_n)$ que lo separa de S y soporta a S. Por la demostración del teorema, sabemos que $b_n = a_n.z$, y puesto que $S \subseteq H_-(a_n, b_n)$,

$$a_n.x \geq a_n.z_n \quad \forall x \in S$$

Lo cual es equivalente a

$$a_n.(x - z_n) \geq 0 \quad \forall x \in S$$

Asumamos, sin perdida de generalidad, que $||a_n|| = 1$. Así, la sucesión $\{a_n\}_n$ dada por $a_n = z - z_n$ está definida sobre la bola unitaria de $\mathbb{R}^n$, que es compacta. Por lo tanto, dicha sucesión tiene una subsucesión convergente que notaremos $\{a_{n_k}\}$. Llamemos $\overline{a}$ a su límite, que también cumple que $||\overline{a}|| = 1$.

Tenemos entonces $\{z_{n_k} \to z\}$ y $\{a_{n_k} \to \overline{a}\}$ y

$$a_{n_k}.(x - z_{n_k}) \geq 0 \quad \forall x \in S$$

Por lo tanto

$$\overline{a}.(x - z) \geq 0 \quad \forall x \in S$$

y

$$\overline{a}.x \geq \overline{a}.z \quad \forall x \in S$$

Con lo cual, si tomamos el hiperplano $H(\overline{a}, b)$ con $b = \overline{a}.z$, es claro que $S \subseteq H(\overline{a}, b)_-$ y, además, $z \in H(\overline{a}, b)$. □

Teorema 4. *Sea $S \in \mathbb{R}^n$ un conjunto no vacío, convexo y cerrado. Entonces S tiene un punto extremo $\iff$ S no contiene una recta.*

Demostración:

- ($\Longrightarrow$) Sea $S \in \mathbb{R}^n$ un conjunto no vacío, convexo y cerrado, y sea $\overline{x}$ un punto extremo de S. Supongamos que S contiene una recta y lleguemos a una contradicción.

  Para que esto pase, debe existir $x \in S$ y $d \in \mathbb{R}^n$ tal que la recta $L = \{x + \lambda d : \lambda \in \mathbb{R}\} \subseteq S$. Para cada $n \in \mathbb{N}$ definamos

  $$x_n = (1 - \frac{1}{n})\overline{x} + \frac{1}{n}(x + n.d)$$

  Notemos que tanto $\overline{x}$ como $x + n.d$ son puntos que están en S (convexo), y como lo anterior es una combinación convexa, se tiene que $x_n \in S \quad \forall n \in \mathbb{N}$. Puesto que S es cerrado y la sucesión está incluida en S, su límite (de existir) también está en S.

  $$\lim_{n\to\infty} x_n = \overline{x} \lim_{n\to\infty} (1 - \frac{1}{n}) + \lim_{n\to\infty} x.\frac{1}{n} + d = \overline{x} + d$$

Pero cambiando el signo y resolviendo análogamente, se llega a que $\overline{x} - d$ también está en S. Esto último implica que entonces $\overline{x}$ no es un punto extremo de S, puesto que

$$\overline{x} = \frac{1}{2}(\overline{x} + d) + \frac{1}{2}(\overline{x} - d)$$

es decir, es una combinación lineal de dos puntos que están en el conjunto. Luego, S no puede contener una recta.

- ($\Longleftarrow$) Queremos ver que si S no contiene una recta, entonces S debe contener un punto extremo. Para probar esto, vamos a verlo por inducción en la dimensión del espacio.

  Supongamos que $n = 1$. Puesto que S es un conjunto en $\mathbb{R}$ que no contiene una recta, debe ser acotado. En consecuencia, tiene ínfimo o supremo. Pero, por hipótesis, S es también cerrado, con lo cual es un mínimo o máximo. Luego, S tiene un vértice.

  Asumamos ahora que $S \subseteq \mathbb{R}^{n+1}$, no contiene una recta y que el resultado vale para $\mathbb{R}^n$. Puesto que S no contiene una recta, debe existir un punto x_0 en la frontera de S. Como estamos bajo las hipótesis del teorema 3, existe un hiperplano soporte de S que contiene a x_0. Sea $H^{x_0} = \{x \in \mathbb{R}^{n+1} \quad | \quad a^T.x = b\}$ para algún $a \in \mathbb{R}^{n+1}$ y algún $b \in \mathbb{R}$ tal que $S \subseteq H_+^{x_0}$. Esto es, $a^T.x \leq a^T.x_0 = b$ (ya que $x_0 \in H^{x_0}$) $\quad \forall x \in S$.

  Analicemos el conjunto $S \cap H^{x_0}$. Este conjunto es cerrado, puesto que tanto S como H^{x_0} lo son. Más aún, es convexo, por la misma propiedad. Dado que $x_0 \in S \cap H^{x_0}$, tenemos que es distinto de vacío. Notemos que $S \cap H_{x_0}$ es un subconjunto de $\mathbb{R}^n$, por lo que podemos aplicar la hipótesis inductiva y afirmar que existe un extremo $\overline{x}$ de dicho conjunto. Pero lo que queremos ver es que S tiene un punto extremo. Bueno, veamos que efectivamente $\overline{x}$ es un vértice de S.

  Supongamos que puedo escribir a $\overline{x}$ como combinación convexa de dos puntos dados x_1, x_2 de S:

  $$\overline{x} = \lambda x_1 + (1 - \lambda)x_2$$

  para algún $\lambda \in (0, 1)$. Multiplicando a todos los términos por a^T obtenemos:

  $$a^T\overline{x} = \lambda a^T x_1 + (1 - \lambda)a^T x_2$$

  Puesto que $x_1, x_2 \in S$ y $\overline{x} \in H_{x_0}$, por las propiedades del hiperplano separador, sabemos que $a^T x_1 \leq a^T\overline{x}$ y $a^T x_2 \leq a^T\overline{x}$. En consecuencia,

  $$a^T\overline{x} = \lambda a^T x_1 + (1 - \lambda)a^T x_2 \leq \lambda a^T\overline{x} + (1 - \lambda)a^T\overline{x} = a^T\overline{x}$$

  Por lo tanto, todas las desigualdades anteriores son igualdades lo que concluye en que $x_1, x_2 \in S \cap H^{x_0}$. Pero puesto que $\overline{x}$ es un punto extremo de ese conjunto, no puede escribirse como combinación convexa de dos puntos del mismo conjunto. Esto es un absurdo que proviene de suponer que es un punto extremo de S. $\square$

Recordemos que en un principio hablamos del conjunto de soluciones factibles de un PL. Nuestro objetivo es obtener resultados que nos simplifiquen encontrar dichas soluciones dado que el conjunto de soluciones factibles cumple con las hipótesis de los teoremas dados. A continuación, veremos el resultado mas importante al que queríamos llegar.

Teorema 5. *Sea $S = \{x \in \mathbb{R}^n \quad | \quad A.x \leq b\}$ y sea el problema de PL P: minimizar el conjunto $\{c^T x : x \in S\}$. Si S tiene un punto extremo y P un óptimo entonces existe un óptimo que también es un punto extremo de S.*

Demostración:
Sea α^* el valor óptimo y consideremos el conjunto de todos los puntos donde se alcanza el óptimo $O = \{x \in S \quad | \quad c^T x = \alpha^*\}$. Por hipótesis, S tiene un punto extremo y por el teorema 4, sabemos que no contiene una recta. Puesto que O es un subconjunto de S, tampoco contiene una recta. Observemos que O cumple con las hipótesis del teorema 4, ya que es distinto de vacío (P tiene solución óptima), es claramente cerrado y convexo. Aplicando nuevamente el teorema, sabemos que O tiene un punto extremo que llamaremos $\overline{x}$. Veamos que es vértice de S.

Repitiendo el razonamiento de la demostración anterior, supongamos que $\overline{x}$ no es un vértice de S, por lo que podemos escribirlo como una combinación convexa de puntos de S. Esto es:

$$\overline{x} = \lambda x_1 + (1 - \lambda) x_2$$

para $x_1, x_2 \in S$ y $\lambda \in (0, 1)$. Multiplicando a ambos lados por c^T:

$$c^T \overline{x} = \lambda c^T x_1 + (1 - \lambda) c^T x_2$$

Puesto que $x_1, x_2 \in S$ y α^* es el valor óptimo de P, debe ocurrir que $c^T x_1 \leq c^T \overline{x} = \alpha^*$ y $c^T x_2 \leq c^T \overline{x} = \alpha^*$. Para que se satisfaga la igualdad anterior, debe pasar que $c^T x_1 = c^T x_2 = c^T \overline{x} = \alpha^*$. Es decir, $\overline{x}$ se escribe como una combinación convexa de puntos de O, pero es un punto extremo de O. Absurdo. Luego, $\overline{x}$ es un punto extremo de S. $\square$

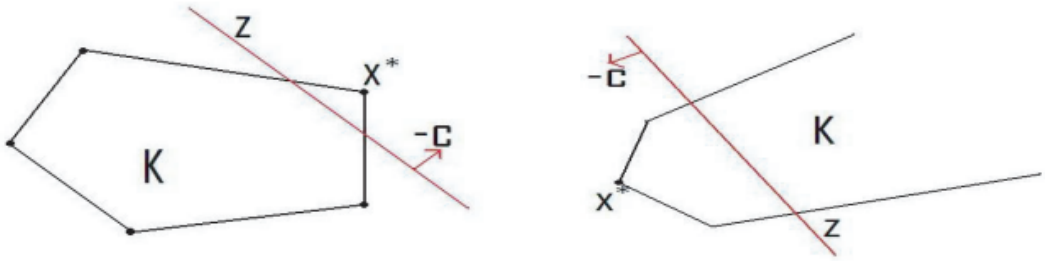
Observación: Si existen dos puntos extremos de S que son óptimos del problema P, entonces todo punto de la combinación convexa de estos también es óptimo.

Sean $x_1, x_2 \in S$ puntos extremos y óptimos de P. Considero $\overline{x} = \lambda x_1 + (1 - \lambda) x_2$ para algún $0 \leq \lambda \leq 1$. Pero multiplicando por c^T a cada término, se tiene:

$$c^T \overline{x} = \lambda c^T x_1 + (1 - \lambda) c^T x_2 = \lambda \alpha^* + (1 - \lambda) \alpha^* = \alpha^*$$

Luego, $\overline{x}$ es óptimo para cualquier λ entre 0 y 1.

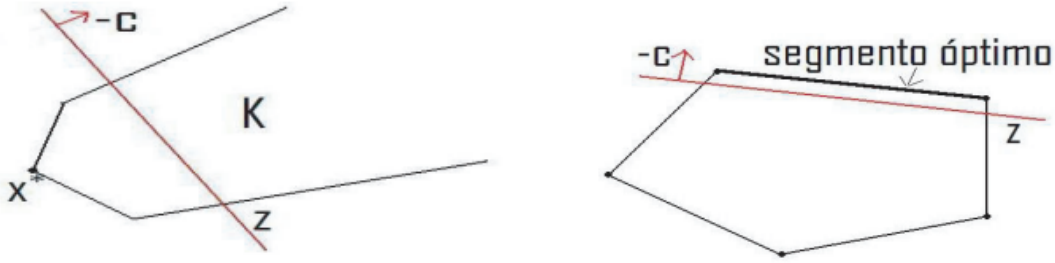
Existen muchos tipos de problemas que dependen de distintos factores: el conjunto de soluciones factibles, la función objetivo, entre otras cosas. Veamos algunos ejemplos de soluciones posibles que pueden encontrarse en problemas de PL. En todas las figuras de 1.3 y 1.4, la recta roja representa el gradiente de la función objetivo



(a) Único óptimo en polítopo

(b) Único óptimo en poliedro

Figura 1.3: Tipos de soluciones



(a) Problema no acotado

(b) Infinitos óptimos en polítopo

Figura 1.4: Tipos de soluciones

y su dirección depende de si el problema minimiza o maximiza. En la figura 1.3a tenemos un polítopo como conjunto de soluciones factibles K y siguiendo la dirección de crecimiento del gradiente es claro que en x^* se alcanza el óptimo. Análogamente, sucede en la imagen 1.3b, que al estar minimizando alcanza un óptimo en K a pesar de ser no acotado. Por lo contrario, en 1.4a el problema es no acotado, dado que la función objetivo puede crecer infinitamente en el conjunto de soluciones factibles. Por último, tenemos el caso de la observación antes dicha, en la que tenemos un segmento completo alcanzando el valor óptimo del problema (1.4b).

Los resultados antes vistos nos proporcionan condiciones muy fuertes sobre los óptimos de un problema así. Por el teorema 5, sabemos que basta con recorrer los vértices del poliedro o polítopo dado por el conjunto de soluciones factibles (siempre y cuando dicho óptimo exista). Pero esto no significa que vaya a ser fácil. Dicho conjunto puede tener un número muy grande de vértices, generando que la evaluación de cada uno sea sumamente costosa. Además, el solo hecho de agregar una variable, aumenta la dimensión del espacio donde vive el conjunto, haciéndolo más difícil aún. Se necesita entonces una "estrategia" para recorrer los vértices de manera inteligente y encontrar el punto óptimo del problema. Aquí es donde entra en juego el algoritmo *simplex*.

1.2. El algoritmo de SIMPLEX

Este algoritmo es el más famoso para resolver este tipo de problemas. Fue creado en el año 1947 por *George Dantzig* [6]. Lo que hace el algoritmo, dado que sabemos que el óptimo (de existir) se encuentra en un punto extremo, es moverse de vértice en vértice siempre que mejora la función objetivo. El criterio de detención del algoritmo será verificar que no existe un punto donde mejore la función objetivo, o bien encontrar una dirección que muestre que el problema es no acotado. Para ver como funciona el algoritmo, recordemos la formulación estándar de un problema de PL:

$$\begin{aligned} Min\ z &= c^T x \\ Sujeto\ a:\ Ax &= b \\ x &\geq 0 \end{aligned}$$

Veamos que todo problema puede formularse de esta forma:

- Si la función objetivo es Max $= z = c^T x$, se puede transformar en Min $= -c^T x$ y luego se multiplica por -1 al resultado final.
- Si las restricciones son de desigualdad, se agregan variables que llamaremos **variables de holgura**:

  $$\begin{aligned} a_i x_1 + ... + a_{in} x_n &\leq b_i \\ a_i x_1 + ... + a_{in} x_n + x_{n+i} &= b_i \quad con \quad x_{n+i} \geq 0 \end{aligned}$$

  Si se tiene $\geq$, se procede igual pero restando la variable de holgura (la cual es siempre no negativa).
- Las variables no positivas o irrestrictas pueden reemplazarse por no negativas:
  - Si tengo $x_k \leq 0$, se reemplaza por $-x_k \geq 0$
  - Si tengo x_k irrestricta, se intercambia por: $x_k = x'_k - x''_k$ con $x'_k, x''_k \geq 0$

  Al resolver el nuevo problema, puedo obtener la solución del problema original efectuando los mismos cambios hacía atrás en las variables.

Recordando que resolver un problema de PL es equivalente a encontrar el vértice con el mejor valor en la función objetivo, vamos a introducir el concepto de solución básica factible y mostrar como caracterizar los puntos extremos del poliedro factible en términos de estas soluciones básicas factibles.

Sea el sistema $Ax = b$, $x \geq 0$ con $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$ y $b \in \mathbb{R}^m$. Se asume que $m \leq n$ y $rang(A) = m$ (si $rang(A) < m$ elimino las filas redundantes).

Denotamos:

- $A_{.j} \longrightarrow$ columna j de la matriz A.

- $A_{i.} \longrightarrow$ fila i de la matriz A.
- $a_{ij} \longrightarrow$ coeficiente de la fila i, columna j de la matriz A.
- $Ax = b \longrightarrow A_{.1}x_1 + ... + A_{.n}x_n = b, \quad x_j \geq j.$

Puesto que $rang(A) = m$, la matriz tiene m columnas linealmente independientes. Sea B la submatriz de A formada por estas columnas. Reordenamos las columnas de A de modo que $A' = [B|R]$, siendo R la matriz residual formada por las $n - m$ columnas restantes que no están en B.

Reordenemos también el vector x transformándolo en $x' = [x_B|x_R]^T$. Entonces:

$$Ax = b \iff [B|R][x_B|x_R]^T = b$$

$$\implies Bx_B + Rx_R = b$$

Como las columnas de B son linealmente independientes, B es invertible.

$$\implies B^{-1}Bx_B + B^{-1}Rx_R = B^{-1}b$$

$$\implies I_d x_B + \overline{R}x_R = \overline{b}$$

siendo I_d la matriz identidad, $\overline{R} = B^{-1}R$ y $\overline{b} = B^{-1}b$

Definición 8. *La solución $x = [x_B|x_R]$, con $x_B = \overline{b} = B^{-1}b$ y $x_R = 0$ se denomina* ***solución básica****, ya que está asociada a una base de vectores columna de la matriz A.*

Si además $x_B \geq 0$, entonces se llama ***solución básica factible*** *puesto que satisface todas las restricciones del problema.*

Dada la definición anterior, notamos a las componente de x_B de x como **variables básicas** y las coordenadas de X_R como **variables no básicas**. Para que una solución básica sea factible se necesita que $\overline{b} \geq 0$. Una matriz básica B que cumple $B^{-1}b$ es no negativo se dice **matriz primal factible**.

Podemos entonces caracterizar a las soluciones básicas factibles en función de los vértices del poliedro factible.

Teorema 6. *Un punto $\overline{x} \in K = \{x \in \mathbb{R}^n \quad | \quad Ax = b, \quad x \geq 0\}$ es solución básica factible si y solo si $\overline{x}$ es un punto extremo del poliedro K.*

Demostración:

- ($\implies$) Supongamos, sin perdida de generalidad, que $\overline{x} = (\overline{x_1}, \overline{x_2}, ..., \overline{x_m}, 0, ..., 0)$ es una solución básica factible. Entonces, $\overline{x_1}A_{.1} + \overline{x_2}A_{.2} + ... + \overline{x_m}A_{.m} = b$, donde $A_{.1}, A_{.2}, ..., A_{.m}$ son m columnas linealmente independientes.

  Supongamos que $\overline{x}$ no es un punto extremo, con lo cual podemos escribirlo como la combinación convexa de dos puntos distintos x^1, x^2 de K. Esto es:

$$\overline{x} = \lambda x^1 + (1 - \lambda)x^2, \quad 0 < \lambda < 1$$

En particular, vale que:

$$\overline{x_j} = \lambda x_j^1 + (1-\lambda)x_j^2, \quad j = 1, ..., m$$

$$0 = \lambda x_j^1 + (1-\lambda)x_j^2, \quad j = m+1, ..., n$$

Puesto que x^1 y x^2 son soluciones factibles, todas sus componentes son ≥ 0. En consecuencia, $x_j^1 = x_j^2 = 0 \quad \forall j = m+1, ..., n$.

Además, satisfacen que $Ax^1 = Ax^2 = b$ y por lo anterior debe ser:

$$x_1^1 A_{\cdot 1} + x_2^1 A_{\cdot 2} + ... + x_m^1 A_{\cdot m} = x_1^2 A_{\cdot 1} + x_2^2 A_{\cdot 2} + ... + x_m^2 A_{\cdot m} = b$$

Pero las columnas $A_{\cdot 1}, A_{\cdot 2}, ..., A_{\cdot m}$ son linealmente independientes, por lo que el sistema tiene solución única. Es decir, $\overline{x} = x^1 = x^2$, cuando supusimos que eran puntos distintos. Luego, $\overline{x}$ debe ser un punto extremo de K.

- ($\Longleftarrow$) Supongamos ahora que $\overline{x}$ es un punto extremo de K y reordeno sus coordenadas de manera tal que las primeras $k (k \geq 1)$ son distintas de cero (si $\overline{x} = 0$, es fácil ver que es una solución básica factible, por lo tanto debe existir dicho k). Entonces,

$$\overline{x_1} A'_{\cdot 1} + \overline{x_2} A'_{\cdot 2} + ... + \overline{x_m} A'_{\cdot k} = b \quad con \quad \overline{x_i} \quad \forall i = 1, ..., k \tag{1.1}$$

siendo $A'_{\cdot j}$ las respectivas columnas de A.

Si los vectores $A'_{\cdot 1}, ..., A'_{\cdot k}$ son linealmente independientes, $\overline{x}$ es una solución básica factible, ya que si $k = m$ tenemos una base de B y si $k < m$ (k no puede ser mayor que m) se pueden extender estas columnas a una base de B.

Asumamos que no son linealmente independientes. Por definición, existen $\alpha_1, \alpha_2, ..., \alpha_k$ no todos nulos tales que

$$\alpha_1 A'_{\cdot 1} + \alpha_2 A'_{\cdot 2} + ... + \alpha_k A'_{\cdot k} = 0 \tag{1.2}$$

Multiplicando 1.2 por $\theta > 0$ y sumando 1.1 se tiene:

$$(\overline{x_1} + \theta\alpha_1) A'_{\cdot 1} + (\overline{x_2} + \theta\alpha_2) A'_{\cdot 2} + ... + (\overline{x_m} + \theta\alpha_k) A'_{\cdot k} = b$$

Por otro lado, haciendo (1.1) - θ(1.2) resulta

$$(\overline{x_1} - \theta\alpha_1) A'_{\cdot 1} + (\overline{x_2} - \theta\alpha_2) A'_{\cdot 2} + ... + (\overline{x_m} - \theta\alpha_k) A'_{\cdot k} = b$$

De esta forma, notar que obtenemos dos soluciones x^1, x^2 para el sistema $Ax = b$, dadas por

$$x^1 = (\overline{x_1} + \theta\alpha_1, ..., \overline{x_k} + \theta\alpha_k, 0, ..., 0)$$

$$x^2 = (\overline{x_1} - \theta\alpha_1, ..., \overline{x_k} - \theta\alpha_k, 0, ..., 0)$$

Para que x^1 y x^2 sean factibles, todas sus coordenadas deben ser no negativas. Pero $\overline{x_j} > 0$, entonces puedo elegir $\theta > 0$ tan pequeño como sea necesario de modo que esto ocurra. Notar que en cada coordenada debe cumplirse que

$$-\frac{\overline{x_j}}{\alpha_j} \geq \theta \quad si \quad \alpha_j < 0$$

$$\frac{\overline{x_j}}{\alpha_j} \geq \theta \quad si \quad \alpha_j > 0$$

Si algún $\alpha_j = 0$, la coordenada es positiva sin importar θ. Si $\alpha_j < 0$, x^2 tiene todas sus coordenadas positivas, por lo que nos concentramos en x^1. Y si $\alpha_j > 0$ es análogo pero invirtiendo los vectores.

Así, elegimos θ tal que $0 < \theta < \overline{\theta}$ con $\overline{\theta} = min\{\theta_1, \theta_2\}$ donde $\theta_1 = \text{mín}_{\alpha_j<0}\{-\frac{\overline{x_j}}{\alpha_j}\}$ y $\theta_2 = \text{mín}_{\alpha_j>0}\{\frac{\overline{x_j}}{\alpha_j}\}$, y para estos θ, x^1 y x^2 son puntos factibles.

Pero, $\overline{x} = \frac{1}{2}x^1 + \frac{1}{2}x^2$, siendo x^1 y x^2 factibles y distintos de $\overline{x}$. Esto contradice la hipótesis de que $\overline{x}$ es un vértice. Luego, $\overline{x}$ debe ser una solución básica factible. □

El algoritmo *simplex* examina vértices del poliedro factible, o sea que, por el teorema anterior, determina soluciones básicas factibles.

Si un vértice dado k no es óptimo (y existe posibilidad de mejora en la función objetivo), el algoritmo genera un nuevo vértice factible **adyacente** al actual. Geométricamente, dos vértices del poliedro factible K son adyacentes si el segmento de recta que los une corresponde a una arista de K. Más formalmente,

Definición 9. *Dos puntos extremos del poliedro factible K, x^1 y x^2, son adyacentes si:*

$$rango(\{A_{\cdot j}/x_j^1 > 0 \vee x_j^2 > 0\}) = |\{A_{\cdot j}/x_j^1 > 0 \vee x_j^2 > 0\}| - 1$$

Esta definición establece que las bases asociadas a puntos extremos adyacentes difieren sólo de un vector.

Modificar la solución básica factible asociada a un vértice x^k de K para obtener la solución asociada a un vértice adyacente a x^k es equivalente a modificar la matriz básica asociada a x^k, donde la nueva matriz básica difiere de la primera en solo un vector columna.

Este procedimiento se itera mejorando la solución de la función objetivo hasta obtener el óptimo. Pero, ¿ Cómo saber si la solución en la que me encuentro es óptima?

Examinaremos la función objetivo para determinar si es posible disminuir su valor modificando el valor de las variables no básicas. Esto se debe a que para pasar a un vértice adyacente, sólo se puede modificar el valor de una variable no básica. Esta idea es tomada como criterio de optimalidad por el algoritmo para terminar.

En resumen, el algoritmo consta de 2 fases:

- Inicialización: Paso en el cual se debe determinar un vértice inicial factible para comenzar a iterar.

- Optimización: Verifica si el vértice es óptimo. Si lo es, termina. Si no busca un vértice factible adyacente al actual tal que la función objetivo mejora.

1.2.1. Complejidad del algoritmo SIMPLEX

¿Es eficiente *simplex* para resolver problemas lineales? Bueno, en general sí, aunque en algunos casos puede ser sumamente ineficiente. Recordemos que *simplex* recorre los puntos extremos de un poliedro y el número de estos puede llegar a ser exponencial en el tamaño del problema, con lo cual, si llegara a recorrer todos ellos, el algoritmo sería muy poco eficiente.

En 1972, Klee y Minty [15] desarrollaron un ejemplo de PL en n variables para el cual *simplex* examina en el orden de $\mathcal{O}(2^n)$ vértices del poliedro antes de llegar al óptimo. Por lo tanto, para dicho caso, el algoritmo tiene un comportamiento muy malo. Sin embargo, en la práctica estos casos son raros y el algoritmo sigue siendo uno de los más competitivos para resolver problemas de PL.

Entonces, ¿existe algún algoritmo que siempre resuelva los problemas de PL en tiempo polinomial? En efecto, si. El primero que aparece se lo conoce como *Método de los elipsoides*, creado por Khachian en 1979. Este algoritmo es teóricamente mejor que *simplex*, en el sentido del peor caso. Sin embargo, en la práctica, en la gran mayoría de los casos *simplex* funciona mucho mejor.

En 1984, Karmarkar [14] dió a conocer un nuevo algoritmo para PL, también de tiempo polinomial, pero esta vez se anunciaba que también funcionaba mejor que *simplex* en la práctica. La característica de este algoritmo es que se mueve por el interior de la región factible. Por esa razón, se lo conoce como el *Método del punto interior*.

A partir de entonces, otros algoritmos de punto interior fueron desarrollados para resolver PL. Pero la conclusión actual es que efectivamente existen métodos mejores que *simplex*, y más aún cuando son problemas de gran tamaño.

¿Cuál es el motivo para utilizar *simplex* habiendo otras opciones mejores?

Estos análisis en general se plantean sobre el peor caso posible. Existen estudios sobre la complejidad promedio del algoritmo *simplex*, asumiendo cierto modelo probabilístico para las posibles instancias de un problema de PL, que demuestran que la complejidad es del orden de $\mathcal{O}(min\{n, m\})$.

1.3. Programación lineal entera y mixta

En muchos casos puede suceder que las variables tomen únicamente valores enteros. Existen muchos ejemplos en donde no podemos elegir una fracción de algo, simplemente es tomarlo o no. Por ejemplo, digamos que tenemos variables representando cuanta cantidad de pelotas comprar. Es claro que podemos comprar 1, 2, 10 pelotas, pero no podemos comprar 2.4 pelotas. Dentro de este tipo de variables tenemos un subconjunto muy importante de variables denominadas *variables binarias*. Como lo dice su definición, estas variables toman sólo los valores 0 y 1. Son muy usuales en modelos donde debemos elegir si realizamos o no cierta acción.

En el caso en que todas las variables sean enteras o binarias, estamos ante un problema de programación entera. Mientras que si el problema tiene tanto variables reales como variables enteras o binarias, se dice que estamos en un problema de programación mixta. Notemos que en estos tipos de problemas, el conjunto de soluciones factibles es un conjunto discreto. Un problema mixto, tiene la siguiente formulación:

$$\begin{aligned} &Minimizar\ c^T x + d^T y \\ &Sujeto\ a : Ax + Dy \leq b \\ &\qquad y_i \in \mathbb{Z}_{\geq 0} \qquad \forall i \in I \subseteq \{1, ..., n\} \\ &\qquad x_j \geq 0 \qquad \forall j \in I^c \end{aligned} \tag{1.3}$$

Notar que si el conjunto de soluciones factibles es acotado, el número de soluciones es finito. Pero, ¿esto significa que el problema es fácil de resolver? No necesariamente, porque el número de soluciones puede ser exponencial en el tamaño del problema. Por ejemplo, el conjunto definido por: $0 \leq x_i \leq 1$ con $x_i \in \mathbb{Z}$ para $i = 1, ..., n$ es un conjunto con 2^n vértices.

Es claro que para estos problemas no podemos aplicar el algoritmo de *simplex*, dado que necesitamos que las variables estén en $\mathbb{R}^n$. Lo que podemos hacer es utilizar algún algoritmo que utilice *simplex* para resolver subproblemas con variables reales y tenga una estrategia para moverse entre las variables enteras.

1.4. Branch and & Bound

Consideremos que tenemos el siguiente problema lineal entero:

$$\begin{aligned} &Minimizar\ z = c^T x \\ &Sujeto\ a : Ax \leq b \\ &\qquad x_i \in \mathbb{N}_0 \qquad \forall i = \{1, ..., n\} \end{aligned} \tag{PE}$$

Definición 10. *El problema lineal continuo que se obtiene del problema (PE) al omitir las restricciones de integridad de las variables se denomina* ***relación lineal*** *de (PE).*

La relajación lineal del (PE) sería:

$$\begin{aligned} &Minimizar\ z = c^T x \\ &Sujeto\ a : Ax \leq b \\ &\qquad x_i \geq 0 \qquad \forall i = \{1, ..., n\} \end{aligned} \tag{P_0}$$

Observación 1. *Si $C = \{x \in \mathbb{N}_0 / Ax \leq b\}$ es el conjunto factible de (PE) y $C_0 = \{x \in \mathbb{R}_{\geq 0} / Ax \leq b\}$ el conjunto factible de (P_0) es claro que $C \subseteq C_0$, por lo que cualquier óptimo de C será factible en C_0 aunque no necesariamente al revés.*

Observación 2. *Si la solución óptima de la relajación lineal de (PE) es entera, entonces esta solución es óptima para (PE).*

La estrategia de enumerar todas las soluciones factibles de un problema entero (y elegir la mejor) se denomina **enumeración explícita**. Como ya mostramos, en muchas ocasiones esto es impracticable. La idea es enumerar de forma "inteligente" de modo que no sea necesario pasar por aquellas soluciones que sabemos que no serán óptimas.

Este es el objetivo de **Branch & Bound** [1]: Intentar una enumeración parcial que asegure pasar a una solución óptima. A esto último se lo llama **enumeración implícita**.

La idea de este algoritmo consiste en particionar el conjunto factible de la relajación del problema en cuestión y resolver los subproblemas resultantes de esta partición (proceso de ramificación - Branch). Cada una de estas resoluciones, podemos resolverla con *simplex*. A partir del análisis de las soluciones obtenidas para los subproblemas y de sus respectivos óptimos (o cotas para los valores óptimos) se puede establecer si es posible obtener mejores soluciones dividiendo nuevamente algún subproblema (proceso de acotamiento - Bound). Si se concluye que cierto subproblema no puede mejorar (por razones que veremos) la solución actual, no se le ramifica. En caso contrario, se continúa ramificando por dicho problema. Este es un pequeño resumen de como funciona el algoritmo.

Veamos ahora el criterio formalmente. Tenemos el problema:

$$\begin{aligned} &Minimizar\ z = c^T x \\ &Sujeto\ a : Ax \leq b \\ &\qquad x_j \in \mathbb{N}_0 \qquad \forall j \in I \subseteq \{1, ..., n\} \\ &\qquad x_i \geq 0 \qquad \forall i \in I^c \end{aligned} \tag{PE}$$

y asumamos que el conjunto factible es acotado.

Sea $R_0 = \{x \in \mathbb{R}^n / Ax \leq b, x_j \geq 0 \quad \forall j\}$ y sea la relajación lineal de PE:

$$\begin{aligned} Minimizar\ z_0 = c^T x \\ Sujeto\ a : x \in R_0 \end{aligned} \tag{P_0}$$

Sea x^0 una solución óptima de P_0 obtenida con *simplex*, es decir $x^0 \in R_0$ y $c^T x^0 = z_0$.

Si $x_j^0 \in \mathbb{N}_0 \quad \forall j \in I \quad PE$.

Si no, existe un índice $k \in I$ tal que x_k^0 no pertenece a $\mathbb{N}_0$.

Usando este valor podemos particionar (R_0) en dos subconjuntos:

$$R_0^+ = R_0 \cap \{x \in \mathbb{R}^n / x_k \geq \lfloor x_k^0 \rfloor + 1\}$$

$$R_0^- = R_0 \cap \{x \in \mathbb{R}^n / x_k \leq \lfloor x_k^0 \rfloor\}$$

obteniendo los siguientes subproblemas de (R_0):

$$\begin{aligned} Minimizar\ z_0^+ &= c^T x \\ Sujeto\ a : x &\in R_0^+ \end{aligned} \tag{P_0^+}$$

$$\begin{aligned} Minimizar\ z_0^- &= c^T x \\ Sujeto\ a : x &\in R_0^- \end{aligned} \tag{P_0^-}$$

Claramente, $z_0^+ \geq z_0$ y $z_0^- \geq z_0$.

Además, si x^* es solución óptima del PE, x^* debe estar en R_0^+ o en R_0^-, pues x_k^* debe ser entera.

A continuación, debemos elegir uno de los subproblemas y resolverlos. Supongamos que tomamos P_0^+, lo resolvemos con *simplex* y obtenemos una solución óptima x^1. Si $x_j^1 \in \mathbb{N}_0 \quad \forall j \in I$, tenemos una solución factible. ¿Es óptima para el problema PE? No necesariamente, ya que en el subproblema P_0^- puede existir una mejor solución. Lo que podemos asegurar es que conseguimos una cota superior del óptimo (ya que minimizamos).

En el caso en que x^1 no es una solución factible para PE, se continúa ramificando a través de las coordenadas no enteras.

De este modo, se construye un árbol donde de cada nodo se derivan dos ramas. Eventualmente, en las ramas finales estarían todas las soluciones factibles de PE.

Sin embargo, la cantidad de ramificaciones puede ser muy grande, por lo que sería conveniente evitar pasar por todos los subproblemas. Dicho eso, existen nodos en los cuales no vale la pena continuar ramificando y se poda dicha rama. Las razones para no ramificar son:

1. El problema en el nodo resulta infactible. Obviamente, cualquier ramificación también lo será.

2. La solución en el nodo es factible para PE. Por lo tanto, cualquier ramificación no podría tener una mejor solución entera. El valor de la función objetivo con dicha solución debe guardarse si es el mejor que tengo hasta el momento. A este valor se lo conoce como **incumbente**.

3. Si en un nodo del árbol el valor óptimo es mayor o igual que el valor incumbente (asumiendo que estamos minimizando), es claro que ramificar dicho nodo solo dará soluciones enteras peores o iguales a la mejor obtenida hasta el momento.

De esta forma, una vez que recorra todos los nodos que pueden darme una mejor solución, vuelvo al nodo donde obtuve el valor incumbente y será mi solución óptima del PE.

El valor incumbente se inicia en $+\infty$ y va mejorando a medida que se consiguen soluciones enteras factibles. Suele ser útil aplicar una heurística al problema,

generalmente que devuelva una "buena"solución entera factible y empezar el valor incumbente con dicho valor.

Una pregunta que suele hacerse al resolver con el algoritmo es ¿cuál variable elijo para ramificar? Existen algunas técnicas que permiten estimar el grado de mejora de la función objetivo que se tendrá al ramificar en cada una de las variables. Otra pregunta frecuente es ¿qué nodo elegir para ramificar? Notar que podemos ramificar a lo ancho o en profundidad. En principio, esta última estrategia garantiza mejorar más rápido la cota para la función objetivo. Igualmente, no hay forma de garantizar que la rama examinada incluya a la solución del problema.

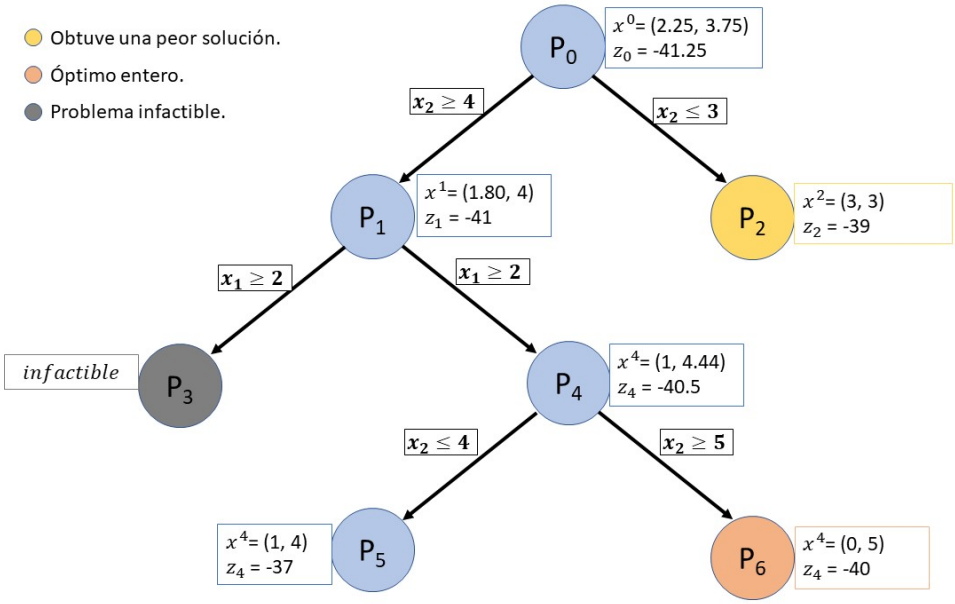


Figura 1.5: Ejemplo de Branch & Bound

En la figura 1.5 podemos ver un pequeño ejemplo en dos dimensiones, donde en el problema inicial la solución tiene ambas coordenadas no enteras (el problema explícito no es de importancia para ilustrar como funciona el algoritmo). Se elige ramificar con x_2 y se resuelve utilizando *simplex* cada subproblema. Podemos elegir cualquiera para comenzar a resolverlos. Iniciando con el problema P_1, la segunda coordenada resulta ser entera, por lo que podemos ramificar únicamente con la primera. Al particionar allí, llegamos a un problema infactible, por lo que conseguimos la primer poda del árbol. Siguiendo con el problema P_4, se vuelve a partir en la segunda variable, obteniendo los subproblemas P_5 y P_6. Al resolver P_5 obtenemos la primer solución factible, ya que ambas coordenadas son enteras. El valor incumbente cambia por primera vez, pasando a valer la solución del problema 5. Al continuar con el problema 6, se obtiene otra solución factible con mejor valor que la anterior, por lo que volvemos a actualizar el valor incumbente. Finalmente, volvemos al problema P_2, el cual también resulta en una solución entera. Puesto que el valor de la función objetivo en dicho punto es peor que el valor incumbente, podemos elegir no ramificar y concluir el problema.

Capítulo 2

The Rest Difference Problem

Este problema se enfoca en minimizar las diferencias de descanso que tienen los equipos en un torneo Round Robin. Esto es, si dos equipos se enfrentan en cierta fecha, que los días de descanso con respecto a la fecha anterior sean lo más cercanos posible. Aquel equipo que tenga mas tiempo de descanso antes del partido, tendrá ventaja sobre su rival. Si bien el objetivo es tener paridad en cuanto a los descansos, cuando se construye un fixture, no pueden pasarse por alto otras restricciones de paridad necesarias, como la cantidad de breaks por equipo, los partidos contra equipos fuertes, entre otras cosas.

2.1. Estado del arte

A lo largo de los años, el Sport Scheduling y armado de torneos fue despertando el interés de muchos investigadores. Hoy en día, existen diversos deportes con formatos de torneos muy distintos entre sí, por lo que los criterios de paridad también lo son.

Un torneo Round Robin, que es el utilizado en el presente trabajo, es aquel en el que cada par de equipos se enfrentan entre sí una única vez. Por lo tanto, si el torneo consta de n equipos, se deberán jugar $n(n-1)/2$ partidos en $n-1$ fechas. En [26] se discute sobre 1-factorización de grafos y su relación con los fixtures. En teoría de grafos, una 1-factorización de un grafo completo es un torneo Round Robin. Un 1-factor es un emparejamiento de vértices en $n/2$ pares. Por eso, las aristas entre dos vértices corresponden a un partido en una ronda.

[4] provee una serie de modelos de programación entera para torneos Round Robin con diversas restricciones de paridad dadas, como lo son partidos prohibidos, tener pocos breaks, respetar las preferencias de días pedidas por los equipos, entre otras.

Otro importante aporte al Sport Scheduling se presenta en [11], donde analizan formatos de competencia y fixtures utilizados en 25 ligas de fútbol de Europa en la temporada 2008-2009. Además, comparan distintos criterios de paridad.

Un método popular, simple pero efectivo, de armar un fixture Round Robin, es el método del círculo. Este método consiste en colocar $n/2$ equipos en una fila y los $n/2$ restantes debajo de forma tal que cada equipo juega contra el equipo que tiene

Round I	Round II	Round III	Round IV	Round V
6 ↔ 1	6 ↔ 5	6 ↔ 4	6 ↔ 3	6 ↔ 2
5 ↔ 2	4 ↔ 1	3 ↔ 5	2 ↔ 4	1 ↔ 3
4 ↔ 3	3 ↔ 2	2 ↔ 1	1 ↔ 5	5 ↔ 4

Figura 2.1: Ilustración del método del círculo para 6 equipos

en frente. En la siguiente fecha todos los equipos salvo uno rotan (este es el equipo que está en alguno de los extremos), generando la segunda fecha. Se puede ver como funciona observando la Figura 2.1.

Sin embargo, el orden en el que se juegan los partidos es de suma importancia. Uno de los criterios más famosos e investigados para el armado de fixtures es la minimización de breaks (por ejemplo, en [7], [25], [8]). Un break ocurre cuando un equipo juega dos fechas consecutivas de local o de visitante.

Otra característica importante por la cual importa el orden de los partidos de un fixture, es el llamado "factor de arrastre". Un claro ejemplo de este caso se da cuando un equipo "débil"se enfrenta a dos o más equipos "fuertes"de manera consecutiva, desgastando mucha energía en el primer encuentro. En [12] Goossens analiza como la fátiga generada en los partidos previos afecta el rendimiento de los jugadores de tenis en un Grand Slam, desde 1992 hasta 2011. Lambrechts demostró en [18] que un fixture generado por el método del círculo genera el máximo número de remanentes.

En ciertos torneos, tiene más relevancia la distancia que deben recorrer los equipos en cada fecha antes que los breaks u otra característica nombrada. En el "Traveling Tournament Problem"[10] (derivado del famoso "Traveling Salesman Problem"), tiene como objetivo encontrar un fixture para un doble Round Robin que minimice la distancia recorrida por los equipos, donde dos equipos no pueden jugar entre sí dos fechas consecutivas y la cantidad de partidos consecutivos de visitante y local tienen cotas superiores.

Un torneo Round Robin con tiempo relajado es aquel en el que hay más fechas del mínimo necesario para jugar todos los partidos. Algunos investigadores se concentraron en estos tipos de torneos en el deporte profesional (como en [17]). Este tipo de torneos puede darse por distintas razones como la poca disponibilidad de estadios o que haya fechas en las que los equipos no pueden jugar. En deportes como el ping pong se prefieren estos torneos, debido a que los breaks no tienen tanta importancia como dejar descansar al equipo cierto tiempo o equilibrar la cantidad de partidos que se juega en cada etapa intermedia de la temporada. El problema de encontrar una distribución equitativa de partidos a lo largo de lapsos de tiempo, o en cierta cantidad de estadios es el problema de encontrar un diseño de torneo equilibrado para minimizar el impacto de desequilibrios en los equipos (ver [3], [13], [22]). El equilibrio puede estar relacionado a la calidad de los estadios donde se juegan los

partidos. Si algunos están en peores condiciones que otros, se busca que todos los equipos jueguen la misma cantidad de veces en dicho lugar.

Finalmente, otro criterio de paridad estudiado para armar fixtures es el de descansos equitativos previos al partido. Esto es, que los equipos que deben enfrentarse lleguen al partido con el mismo (o más cercano posible) descanso con respecto a la fecha anterior. Este problema es el que nos concierne en dicho trabajo.

En [16] Knust estudia un torneo single Round Robin que se juega únicamente en un estadio. Los equipos debían viajar al estadio en cada fecha y jugar dos partidos. En dicho trabajo se construye un fixture que minimiza el número de largas esperas entre los partidos y el tiempo total de espera simultáneamente para cualquier número impar de equipos, ya que los equipos preferían no tener largas esperas.

Esta discordancia entre los descansos tiene mucha influencia en los torneos infantiles. Sanchez ([21]) mostró que los chicos que tienen menos tiempo de descanso tienen rendimientos más bajos en cuanto a las distancias recorridas y velocidad de movimiento en torneos en los que juegan varios partidos en un día.

Suksompong investigó torneos Round Robin donde todos los partidos se juegan en distintos momentos priorizando tres factores: garantizar tiempo de descanso, el índice de diferencias en los partidos jugados y el índice de diferencias de descanso ([23]). El índice de diferencias en los partidos jugados de un torneo asincrónico es el número entero p más chico tal que en cualquier punto del calendario, la diferencia entre el número de partidos jugados entre cualesquiera dos equipos es p. El índice de diferencias de descanso es igual a la máxima diferencia en los descansos entre dos equipos.

El problema de minimizar los descansos se divide principalmente en dos subproblemas: "The rest mismatch problem" y "The rest difference problem". El primer criterio fue introducido por Atan y Cavdaroglu en [2]. Se dice que ocurre un "rest mismatch" cuando dos equipos que se enfrentan tienen diferentes tiempos de descanso al llegar al enfrentamiento. Por lo tanto, "The rest mismatch problem" consta de minimizar las ocurrencias de diferencias en el descanso. Por otro lado, "The rest difference problem" tiene en cuenta la diferencia de tiempo que hay en los descansos de los equipos. En [9] se presentan dos modelos, los cuales fueron aplicados para el caso real a la primera división de fútbol de Ecuador: el primero para asignar los partidos a las fechas y el segundo consta de asignar días a cada partido minimizando la diferencia en los descansos. En un trabajo similar, en [5] se investiga el problema de las diferencias en los descansos para un fixture dado, probando que el problema se puede descomponer en la optimización de las rondas por separado y cada descomposición es una instancia de un problema cuadrático de asignación.

2.2. The Rest Mismatch Problem

En el mundo del deporte, una diferencia en los descansos de los oponentes de un partido es un factor determinante para el rendimiento de los equipos. A esta ocurrencia se la conoce como *rest mismatch*. Introducimos así el llamado *The Rest Mismatch Problem*, problema en el cual se busca minimizar el número de *rest mismatches* en un torneo Round Robin. Observar que en este caso, únicamente se toma en cuenta la ocurrencia de esta diferencia y no cuanta es la diferencia de tiempo en sí.

A modo ilustrativo, veamos un ejemplo simple de un torneo Round Robin para $n = 8$ equipos, donde los partidos se juegan todos en un único estadio. Se asume que el torneo es limitado en el tiempo, lo que significa que cada equipo juega un partido por fecha. El número total de partido debe ser $n(n-1)/2 = 28$ a jugarse en 7 fechas (4 partidos en cada una). Por último, se asume que el estadio soporta 2 partidos por día, por lo que cada fecha se lleva a cabo en dos días consecutivos. En el cuadro 2.1 tenemos un fixture representativo de dicho torneo.

Fecha	Día 1(Partido 1)	Día 1(Partido 2)	Día 2(Partido 1)	Día 2(Partido 2)
1	1 vs 2	3 vs 8	4 vs 6	5 vs 7
2	1 vs 3	2 vs 4	6 vs 7	5 vs 8
3	1 vs 4	2 vs 3	5 vs 6	7 vs 8
4	3 vs 4	6 vs 8	2 vs 5	1 vs 7
5	1 vs 5	3 vs 6	2 vs 7	4 vs 8
6	1 vs 6	2 vs 8	3 vs 5	4 vs 7
7	1 vs 8	2 vs 6	3 vs 7	4 vs 5

Cuadro 2.1: Torneo de 8 equipos

Asumiendo que el torneo se juega en días consecutivos, en el cuadro 2.2 se puede ver calculado los días de descanso de cada equipo al llegar al partido.

Podemos notar, viendo el cuadro 2.2 que en el segundo partido del día 1 de la fecha 2 (2 vs 4) los equipos llegan con distintos descansos (el equipo 2 con dos días y el equipo 4 con tan solo uno).

Fecha	Día 1(Partido 1)	Día 1(Partido 2)	Día 2(Partido 1)	Día 2(Partido 2)
1	-	-	-	-
2	2-2	2-1	2-2	2-3
3	2-2	2-2	2-2	2-2
4	2-2	1-1	3-2	3-2
5	1-1	2-2	2-2	3-3
6	2-2	1-1	3-3	2-2
7	2-2	2-2	2-2	2-2

Cuadro 2.2: Diferencias de descansos de los equipos

Actualmente no hay mucha literatura sobre "*The Rest Mismatch Problem*". Sin embargo, en [2] los autores obtuvieron interesantes resultados a presentar a continuación. Vamos a asumir que tenemos un torneo Round Robin de n (par) equipos donde en cada fecha, los $n/2$ partidos a jugarse se deben asignar a k períodos. Dichos períodos pueden ser tanto días como horarios. Además, se asume que la cantidad de partidos que se juegan por período se mantiene igual a lo largo del torneo. Consideramos que existe un *mismatch* cuando el descanso con el que llegan los equipos al partido es distinto. Por lo tanto, no pueden existir *mismatches* en la primer fecha del torneo. Observar que esta diferencia ocurre cuando dos equipos que se enfrentan en cierta fecha j, jugaron en distintos períodos en la fecha $j-1$. El problema consta de encontrar el fixture para n equipos y k períodos con la menor cantidad de *mismatches* posibles. Se nota a dicho problema como $RMP(n,k)$. El caso en el que $k=1$ se obtiene trivialmente que no existen diferencias en los descansos. Por último, notar que 2.1 es un ejemplo de un $RMP(8,2)$.

Los autores presentan en el trabajo dos modelos diferentes de programación lineal entera para el RMP. Además, formulan el problema como un problema de *constraint programming*.

Constraint Programming (CP) es una técnica popular que suele usarse en el mundo del Sportscheduling. Un problema de CP consta de un conjunto de variables (cada una con un dominio finito) y un conjunto finito de restricciones. Cada restricción de un problema CP restringe simultáneamente el valor que las variables pueden tomar. Usualmente este tipo de problemas no tienen función objetivo, ya que el propósito es encontrar una solución factible.

En el caso de RMP, para resolver el problema como un problema de CP, se intenta construir un fixture en el cual el número total de *mismatches* es 0.

A continuación, se presentan una serie de resultados teóricos para el $RMP(n,k)$ obtenidos en [2].

Proposición 1. *En un fixture donde hay un único período con un solo partido, van a existir dos mismatches en cada ronda. Por eso, el número total de mismatches en el torneo va a ser* $2(n-2)$*. Si el número de períodos con un solo partido es* $s, s > 1$*, el mínimo número de mismatches en una ronda es al menos* s*. Entonces, el número total en todo el torneo es al menos* $s(n-2)$*.*

Demostración:
Siempre que haya un período con un partido solo, en la siguiente fecha van a producirse dos *mismatches* debido a que los oponentes de los equipos de dicho partido habrán jugado en un período distinto. Puesto que en la primera fecha no puede haber diferencias de descanso, se sigue entonces que el número de *mismatches* en todo el torneo es 2(n-2). En el caso en el que el número de períodos con un solo partido sea s con $s > 1$, se puede emparejar a cada equipo de un partido de los períodos con un partido con otro equipo de un partido de estos períodos, generando así s *mismatches*. En consecuencia, si esto pasa en todas las fechas salvo la primera, el torneo tiene al menos $s(n-2)$ *mismatches*. □

Corolario 1. *No existe un fixture con cero mismatches cuando el número de equipos es menor a 8 y* $k \geq 2$*.*

Demostración:
Al haber menos de 8 equipos, necesariamente debe existir un período con un único partido. Por lo anterior, cada uno de estos genera *mismatches* en la siguiente ronda. □

Proposición 2. *En caso de existir, el número de mismatches en una solución factible del* $RMP(n,k)$ *debe ser al menos 2.*

Demostración:
Supongamos que ocurre un *mismatch* entre dos equipos. Estos pertenecían a distintos períodos en la fecha pasada. El número restante de equipos en cada uno de estos períodos es impar. Por lo tanto, no todos los equipos de estos períodos pueden jugar la próxima fecha con equipos del mismo período, generando así otro *mismatch*. □

Corolario 2. *El número de mismatches en cualquier ronda de una solución factible de* $RMP(n,2)$ *no puede ser impar.*

Demostración:
Al ser 2 períodos, si hay un *mismatch* entre dos equipos, por lo anterior debe existir un segundo *mismatch* □

Corolario 3. *El número total de mismatches en una solución factible de* $RMP(n,2)$ *nunca puede ser impar.*

Demostración:
Puesto que el número de *mismatches* en cada ronda es par por el corolario 2, el número total también debe ser par. □

2.2.1. Heurística para RMP(n,k)

Estos simples, pero útiles, resultados sirven de base para una heurística propuesta por Atan y Çavdaroğlu en [2]. La misma es aplicable al problema $RMP(n, 2)$ donde n es divisible por 4 y el número de partidos en cada período es igual. El fixture generado por la heurística asegura que el número total de *mismatches* es $n-8\lfloor n/8 \rfloor$. Es decir, que el número de *mismatches* es 4 si n es divisible por 4 pero no por 8, y si es un número divisible por 8, la heurística encuentra un fixture con 0 *mismatches*.

La heurística se divide en dos pasos. El primero consta de crear un fixture inicial con n *mismatches*. En dicho fixture inicial, los *mismatches* están agrupados en dos fechas específicas, que llamaremos $fechas - mismatches$. En el segundo paso, se eliminan todos los *mismatches* de cada $fecha - mismatch$ cambiando el período de algunos partidos específicos de la fecha anterior.

Paso 1: Fixture inicial

Para crear el fixture inicial se hace uso del método del círculo. En la primera fecha, los $n/2$ partidos se juegan de la siguiente manera: 1 vs 2, 3 vs 4, ..., $n-1$ vs n. Se toman los primeros $n/4$ partidos y se asignan al primero de los dos períodos, mientras que los restantes se juegan en el segundo período. Entonces, los partidos de las siguientes $n/2-2$ fechas se juegan de acuerdo al método del círculo. Esto es, para cada período, se ubican los equipos en dos filas de $n/4$ equipos de manera tal que los equipos quedan alineados dos a dos. La primera fecha corresponde a los partidos dichos anteriormente. Los partidos de las fechas restantes se obtienen manteniendo fijo al equipo de arriba a la izquierda y rotando a los otros en el sentido de las agujas del reloj. Estas rotaciones se realizan en los dos períodos simultáneamente, como se puede ver en la imagen 2.2 para un torneo de 16 equipos. Puesto que el método del círculo se aplica a equipos que pertenecen al mismo período, durante todas estas fechas no se generan *mismatches*.

A continuación, se intercambian la mitad de los equipos del primer período con la mitad de los equipos del segundo período. Esto se hace de la siguiente forma: se toman los $n/4$ equipos con valores más chicos de cada período y se los enfrenta (por ende los más grandes se enfrentarán también). Así, los partidos de la fecha $n/2$ son: 1 vs $n/2+1$, 2 vs $n/2+2$, ..., $n/4$ vs $3n/4$ (primer período) y $n/4+1$ vs $3n/4+1$, $n/4+2$ vs $3n/4+2$, ..., $n/2$ vs n (segundo período).

Una vez cambiados los $n/4$ equipos, se aplica una variante del método del círculo a ambos períodos por separado. La misma consiste de dejar la fila de arriba fija (equipos 1 a $n/4$ para el primer período y $n/4+1$ a $n/2$ para el segundo) y rotar únicamente los equipos de la fila de abajo de acuerdo a las agujas del reloj. La cantidad de rotaciones posibles son $n/4-1$. Siguiendo con el ejemplo del torneo con 16 equipos, se puede ver en la imagen 2.3 como funciona este paso.

Observar que al realizar los intercambios de equipos, la primera fecha resulta tener $n/2$ *mismatches*.

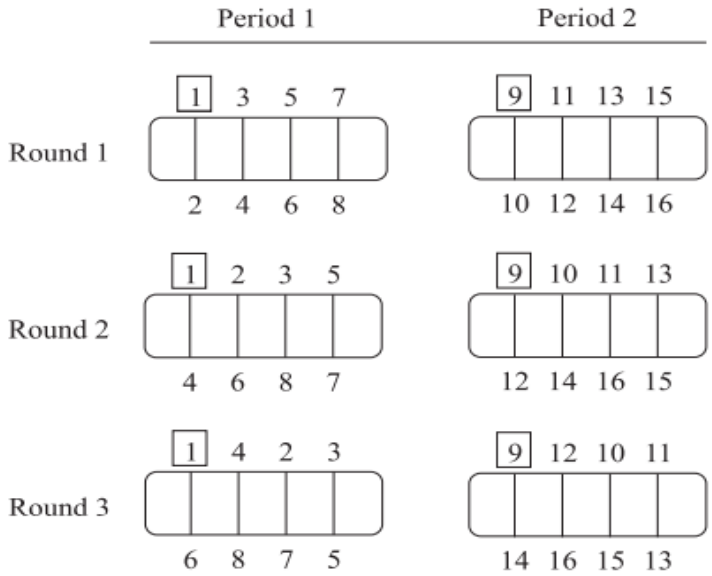


Figura 2.2: Primeras 3 fechas para torneo de 16 equipos

Finalmente, para generar las $n/4$ fechas restantes, se intercambian los equipos 1, ..., $n/4$ del primer período con los equipos $n/4+1$, ..., $n/2$ del otro. De esta manera, se genera la última combinación posible entre los 4 grupos de $n/4$ equipos. Basta con volver a aplicar la variante del método del círculo para finalizar el fixture. Al igual que lo sucedido en el anterior intercambio, se generan $n/2$ *mismatches*, dando un total de n para todo el fixture.

En resumen, generamos un fixture de $n-1$ fechas en tres etapas. En la primera se construyen las primeras $n/2-1$ fechas. En la segunda, luego de un intercambio de equipos en los períodos, se generan las siguientes $n/4$ fechas. En la tercer etapa, aplicando un cambio similar al de la etapa 2, se obtienen las rondas restantes. Puesto que solo se generan *mismatches* al pasar equipos de un período al otro, se obtienen dos $fechas - mismatches$ con $n/2$ *mismatches* cada una.

Paso 2: Eliminando los mismatches

El fixture inicial construido en el paso 1 tiene todos los *mismatches* en las fechas $n/2$ y $3n/4$. Cambiando el orden de las primeras $n/2-1$ fechas no causamos ningún *mismatch* adicional, ya que los equipos se mantienen en el mismo período. Intercambiamos las posiciones de la primera fecha con la $(n/2-1)$-ésima fecha.

Continuando con el ejemplo de 16 equipos, podemos ver en 2.3 como resultan

Fecha	Período 1				Período 2			
7	1-2	3-4	5-6	7-8	9-10	11-12	13-14	15-16
8	1-9	2-10	3-11	4-12	5-13	6-14	7-15	8-16

Cuadro 2.3: Luego del intercambio de fechas en fecha 7

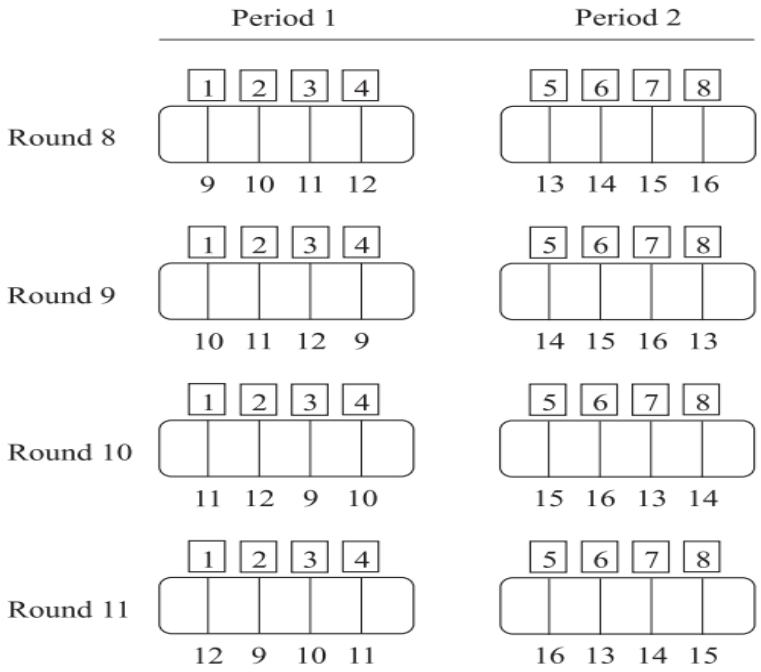


Figura 2.3: 4 fechas siguientes al intercambio de equipos

las fechas $(n/2-1)$ y $n/2$. Si un partido no tiene un *mismatch*, cambiando su período sólo afecta los descansos de los equipos correspondientes, pero no genera un *mismatch*. Basándose en este hecho, en la fecha $(n/2-1)$, intercambiamos los primeros $\lfloor n/8 \rfloor$ del primer período con los últimos $\lfloor n/8 \rfloor$ del segundo período. Con estos cambios, se eliminan los *mismatches* que existían en la fecha $n/2$. En el torneo de 16 equipos (elegido por ser múltiplo de 8), las fechas resultan como en el cuadro 2.4.Se puede ver que no quedan *mismatches* en la fecha $n/2$ como solía haber.

Fecha	Período 1				Período 2			
7	13-14	15-16	5-6	7-8	9-10	11-12	1-2	3-4
8	1-9	2-10	3-11	4-12	5-13	6-14	7-15	8-16

Cuadro 2.4: Luego del intercambio de equipos en fecha 7

Luego, resta eliminar los *mismatches* de la otra $fecha - mismatch$, la fecha $3n/4$. Similarmente, intercambiamos toda la fecha $n/2$ con la $3n/4-1$. Este cambio no genera *mismatches*, ya que desde la fecha $n/2$ hasta la $3n/4-1$ los equipos juegan en un período siguiendo la variante del método del círculo. Para un torneo con 16 equipos, dicho cambio se puede ver en el cuadro 2.5. Repitiendo el paso anterior, en la fecha $3n/4-1$, intercambiamos los primeros $\lfloor n/8 \rfloor$ partidos impares del primer período (en el orden dado) con los primeros $\lfloor n/8 \rfloor$ partidos pares del segundo período. Con dichos cambios, se eliminan todos los *mismatches* de la fecha $3n/4$, y por lo tanto, todos los que había en el fixture inicial. Para concluir el ejemplo que teníamos, en el cuadro 2.6 se ve este último paso para 16 equipos.

Fecha	Período 1				Período 2			
11	1-9	2-10	3-11	4-12	5-13	6-14	7-15	8-16
12	1-13	2-14	3-15	4-16	5-9	6-10	7-11	8-12

Cuadro 2.5: Luego del intercambio de fechas en fecha 11

Fecha	Período 1				Período 2			
11	6-14	2-10	8-16	4-12	5-13	1-9	7-15	3-11
12	1-13	2-14	3-15	4-16	5-9	6-10	7-11	8-12

Cuadro 2.6: Luego del intercambio de equipos en fecha 11

Si el número de equipos es múltiplo de 4 pero no de 8, estas reglas de intercambios siguen valiendo. Sin embargo, en dicho caso, en vez de $n/8$, se deben realizar $(n-4)/8$ intercambios para cada $fecha-mismatch$. Esto eliminaría todos los los $mismatches$ salvo uno por cada $fecha - mismatch$. Finalmente, podemos ver en el cuadro 2.7, el fixture final dado por la heurística para 16 equipos y 0 $mismatches$.

Fecha	Período 1				Período 2			
1	1-3	5-2	7-4	8-6	9-11	13-10	15-12	16-14
2	1-4	2-6	3-8	5-7	9-12	10-14	11-16	13-15
3	1-6	4-8	2-7	3-5	9-14	12-16	10-15	11-13
4	1-8	6-7	4-5	2-3	9-16	14-15	12-13	10-11
5	1-7	8-5	6-3	4-2	9-15	16-13	14-11	12-10
6	1-5	7-3	8-2	6-4	9-13	15-11	16-10	14-12
7	13-14	15-16	5-6	7-8	9-10	11-12	1-2	3-4
8	1-12	2-9	3-10	4-11	5-16	6-13	7-14	8-15
9	1-10	2-11	3-12	4-9	5-14	6-15	7-16	8-13
10	1-11	2-12	3-9	4-10	5-15	6-16	7-13	8-14
11	6-14	2-10	8-16	4-12	5-13	1-9	7-15	3-11
12	1-13	2-14	3-15	4-16	5-9	6-10	7-11	8-12
13	1-14	2-15	3-16	4-13	5-10	6-11	7-12	8-9
14	1-15	2-16	3-13	4-14	5-11	6-12	7-9	8-10
15	1-16	2-13	3-14	4-15	5-12	6-9	7-10	8-11

Cuadro 2.7: Fixture final para 16 equipos

Para finalizar, se obtienen dos resultados que clarifican los intercambios dichos anteriormente.

Proposición 3. *Si todos los partidos de una fecha son mismatches, cambiando el período de un partido en la fecha anterior se eliminan dos mismatches de una* $fecha - mismatch$ *en un* $RMP(n, 2)$.

Demostración:
Sea f una $fecha - mismatch$. Sean $P^f_{i,j}$ y $P^f_{k,l}$ partidos entre i y j, y k y l en la fecha f respectivamente. Sea $P^{f-1}_{i,k}$ el partido de i vs k en la fecha $f-1$. Sabemos que i y j jugaron, en la fecha $f-1$, en distintos períodos, debido a que en la fecha f existe un *mismatch* en su partido. Si cambiamos el período del partido $P^{f-1}_{i,k}$, eliminaríamos dicho *mismatch*. El partido $P^f_{k,l}$ también debe tener un *mismatch* por estar en la fecha f. Esto significa que k y l jugaron sus respectivos partidos en diferentes períodos en la fecha $f-1$. Por eso, cambiando el período del partido $P^{f-1}_{i,k}$se debe eliminar el *mismatch* del partido $P^f_{k,l}$. Como consecuencia, se eliminaron dos *mismatches* de una $fecha - mismatch$ cambiando el período de un partido de la fecha anterior. □

Proposición 4. *Se pueden eliminar $4.\lfloor n/8 \rfloor$ mismatches de cada $fecha-mismatch$ en un $RMP(n,2)$ aplicando las reglas de intercambio del Paso 2: Eliminando los mismatches.*

Demostración:
Sea f una $fecha - mismatch$. Asumamos que n (número de equipos) es múltiplo de 8, $n \geq 8$. Sean Pe^f_1, Pe^f_2, Pe^{f-1}_1 y Pe^{f-1}_2 denotando el primer y segundo período de las fechas f y $f-1$, respectivamente. Notar que hay $n/2$ partidos por fecha y $n/4$ por período. Consideramos un intercambio al movimiento de un partido del Pe^{f-1}_1 al Pe^{f-1}_2, y uno del Pe^{f-1}_2 al Pe^{f-1}_1. Por la proposición 3, al cambiar el período de un partido de la fecha $f-1$, se eliminan dos *mismatches*. Por eso, al hacer un intercambio se eliminan 4 *mismatches*. Para eliminar los $n/2$ *mismatches* de la fecha, se necesitan $n/8$ intercambios. Elegimos $n/8$ partidos del Pe^{f-1}_1 y los movemos al Pe^{f-1}_2. Esta acción elimina $2.n/8$ *mismatches* de la fecha f. Para que los períodos de la fecha $f-1$ queden balanceados, movemos $n/8$ partidos del Pe^{f-1}_2 al Pe^{f-1}_1. Sin embargo, estos últimos partidos no pueden involucrar a los oponentes (en la fecha f) de los equipos del primer movimiento, porque cambiar los períodos de dichos equipos volvería a generar los *mismatches* recién eliminados. La cantidad de estos oponentes "prohibidos" es $2.n/8$. Observar que si los equipos que participan del cambio del Pe^{f-1}_1 al Pe^{f-1}_2 se enfrentarían en la fecha f, no existiría un *mismatch* en su partido, pero f es una $fecha - mismatch$. Por lo tanto, los oponentes prohibidos aparecen en $n/4$ distintos partidos en la fecha f. Para elegir $n/8$ equipos del Pe^{f-1}_2, que no sean estos oponentes, debemos asegurarnos que los oponentes prohibidos jueguen entre sí en $n/8$ partidos en Pe^{f-1}_2. Entonces, los restantes $n/8$ partidos de dicho período no contendrían a los oponentes de los primeros equipos movidos. En base a esto, el problema se reduce a encontrar un conjunto de $n/8$ equipos del Pe^{f-1}_1 y $n/8$ equipos del Pe^{f-1}_2 cuyos $n/4$ equipos se enfrentan en la fecha f. Al hacer estos $n/8$ intercambios, los *mismatches* de la fecha f quedarían eliminados.

En el Paso 2: Eliminando los mismatches, al intercambiar los partidos de la fecha 1 y $n/2-1$, obtenemos los partidos de la fecha $n/2-1$ de manera tal que los equipos de los primeros $n/8$ partidos del primer y segundo período jueguen entre sí en $n/4$ partidos distintos de la fecha $n/2$. En este fixture logramos eliminar todos los *mismatches* de esta fecha al intercambiar los primeros $n/8$ partidos del

primer período con los últimos $n/8$ partidos del segundo período de la fecha $n/2-1$. Similarmente, intercambiando los partidos de las fechas $n/2$ y $3n/4-1$, los partidos de la $3n/4-1$-ésima fecha quedan posicionados de tal forma que los equipos de los partidos impares del primer y segundo período, se enfrentan en $n/4$ partidos de la siguiente fecha. Por lo tanto, siguiendo lo hecho en la heurística, intercambiamos los partidos impares del primer período con los pares del segundo de la fecha $3n/4-1$. De esta forma, se eliminan los *mismatches* de la fecha $3n/4$.

Si el número de equipos es un múltiplo de 4, pero no de 8, las reglas de intercambios antes explicadas garantizan eliminar $(n-4)/2$ *mismatches* luego de $(n-4)/8$ cambios de partidos. Es decir, quedan dos *mismatches* sin eliminar, cuando n es divisible por 4 pero no por 8. El número de *mismatches* eliminados puede ser generalizado a $4.\lfloor n/8 \rfloor$. $\square$

2.3. The Rest Difference Problem

Al hablar de descansos en las competencias deportivas, otro de los problemas estudiados es el llamado *The Rest Difference Problem* estudiado en [5] y [24]. Este problema es prácticamente igual al de *The Rest Mismatch Problem* con la salvedad de que en este caso es importante contabilizar cuanta es la diferencia entre los descansos. Recordemos que en el problema anterior únicamente interesaba el hecho de existir una diferencia, mientras que para este problema nos interesa saber (si existe) cuanto fue dicha diferencia. Este problema es el que nos compete en dicho trabajo.

La optimización consta de minimizar la suma de las diferencias en los descansos de todos los partidos de un torneo Round-Robin. Este problema puede presentarse principalmente de dos formas. La primera es crear un fixture (con las restricciones adecuadas) y asignar los partidos a cada período (día u horario) en el mismo modelo. La segunda, es simplemente la asignación de partidos a los períodos de un fixture ya existente o creado en otro modelo anteriormente.

Para ilustrar este problema, podemos considerar un torneo Round-Robin de 8 equipos con dos períodos, cuyo fixture viene dado por el cuadro 2.1 (el mismo dado para RMP). Más aun, en el cuadro 2.2 podemos ver cuantos descansos tiene cada equipo de una fecha a otra. Calculando las diferencias que hay entre los descansos de todos los partidos observamos que este fixture tiene tan solo 4 diferencias (en la fecha 1, el partido 2 de cada día y en la fecha 4 los dos partidos del día 2).

Para comenzar a ver este problema en profundidad, voy a presentar algunos resultados reportados en [24]. En dicho trabajo se describe un modelo de programación mixta-entera que formula el problema en cuestión. Dicho modelo decide en que fecha y que período se debe jugar cada partido minimizando la suma total de las diferencias de descansos. Además, proponen un método exacto de tiempo polinomial para algunas instancias especiales del problema y una matheurística (conexión entre una metaheurística y la programación matemática) que resuelve el problema general.

Supongamos que tenemos un Single Round-Robin con n(par) equipos. Recordemos que un torneo así tiene $n-1$ fechas con $n/2$ partidos cada una. Estos últimos se juegan en p períodos consecutivos. Vamos a denotar γ_d el número de partidos en el período $d \in \{1, 2, ..., p\}$. La diferencia de descansos se define como la diferencia entre períodos de descanso entre los equipos que se enfrentan. *The Rest Difference Problem* $RDP(n, p|\gamma_1, ..., \gamma_p)$ arma un fixture que determina las fechas y los períodos de cada partido tal que la diferencia de descansos total es minimizada.

2.3.1. Resultados teóricos para el problema con períodos con un partido

Sea $p_1, p_1 \geq 0$, denotando el número de períodos con un solo partido. A modo ilustrativo, consideremos el problema $RDP(8, 3|2, 1, 1)$ presentado en el cuadro 2.8 donde $p_1 = 2$. Consideremos, por ejemplo, el partido 1vs4 que se juega en la fecha 3, en el período 3. Puesto que ambos equipos juegan contra distintos oponentes en

Fecha	Período 1		Período 2	Período 3
1	3-6	4-5	2-8	1-7
2	2-7	3-5	1-8	4-6
3	5-7	6-8	2-3	1-4
4	2-4	6-7	1-3	5-8
5	3-8	4-7	2-6	1-5
6	1-6	2-5	7-8	3-4
7	3-7	4-8	1-2	5-6

Cuadro 2.8: Fixture de $RDP(8, 3|2, 1, 1)$

la ronda 4 y que el período 3 solo tiene un partido, se van a generar diferencias en los descansos de la siguiente fecha. Por eso, cuando tenemos este tipo de períodos (es decir, $p_1 > 0$), la suma de las diferencias totales será siempre positiva.

Teorema 7. *La diferencias de descansos en cada fecha es al menos* $p_1 + mod(p_1, 2)$. *Por eso,* $(n-2).[p_1 + mod(p_1, 2)]$ *es una cota inferior para el valor de diferencias totales de un single Round-Robin.*

Demostración:
Consideremos primero el caso en que p_1 es par. Cuando $p_1 = 0$, la cota inferior es trivialmente 0. Asumamos que $p_1 > 0$. Supongamos que d es un período de un solo partido. Como los equipos que juegan en d en la fecha f deben jugar contra otros dos equipos (e y e') en otros dos partidos (p y p') de la siguiente fecha, es inevitable que haya diferencias de descanso en p y p'. El valor de las diferencias de g y g' van a ser 1, que es el menor valor posible, si e y e' juegan entre sí en otro período de un solo partido (digamos d^*) de la fecha f y si d y d^* son consecutivos. De este modo, cada par de períodos de un partido en cada fecha genera al menos 2 diferencias de descanso. En cada fecha, $p_1/2$ pares de períodos de un partido provocan al menos p_1 diferencias. Como las diferencias en los descansos pueden suceder en todas las fechas salvo la primera, el valor de diferencias totales en el torneo debe ser al menos $(n-2).p_1$, cuando p_1 es par.

Consideremos ahora que p_1 es impar. $(p_1 - 1)/2$ pares de períodos de un partido incrementan el número de diferencias totales en $p_1 - 1$. Los equipos del período de un partido restante (llamémoslo d) deben jugar contra otros dos equipos (e y e') en otros dos partidos (p y p') de la siguiente fecha. Puesto que d no puede ser emparejado con otro período de un partido, los partidos p y p' van a generar una diferencia de al menos 1 cada uno. Por lo tanto, p_1 períodos de este tipo en cada fecha van a generar $p_1 - 1 + 2 = p_1 + 1$ diferencias totales en los descansos. Nuevamente, como las diferencias ocurren en $n-2$ fechas, el total de las diferencias del torneo va a ser al menos $(n-2).(p_1 + 1)$ cuando p_1 es impar.

Teniendo en cuenta que, si p_1 es par $mod(p_1, 2) = 0$ y p_1 es impar $mod(p_1, 2) = 1$, se tiene la cota inferior deseada. □

En el cuadro 2.8, vemos un ejemplo cuando p_1 es par. En el cuadro 2.9, un ejemplo cuando p_1 es impar. En ambos fixtures el valor de las diferencias totales es igual a la cota inferior del teorema.

Fecha	Período 1		Período 2	Período 3	Período 4
1	2-7	6-8	3-10	1-9	4-5
2	1-5	7-8	2-6	4-10	3-9
3	1-8	3-4	2-10	6-9	5-7
4	2-9	4-8	5-6	7-10	1-3
5	2-4	8-9	3-5	6-7	1-10
6	1-7	4-9	2-8	3-6	5-10
7	2-5	3-8	1-4	7-9	6-10
8	9-10	5-8	1-6	2-3	4-7
9	1-2	8-10	3-7	4-6	5-9

Cuadro 2.9: Fixture de $RDP(10, 4|2, 1, 1, 1)$

Teorema 8. *Supongamos que tenemos un fixture óptimo para un $RDP(n, p)$ con $RD = LB$, siendo RD la suma de las diferencias totales del torneo y LB la cota del Teorema 7. Más aun, asumamos que p_1 es par y existe un período d con $\gamma_d > 2$ en $RDP(n, p)$. Si d se divide en dos períodos consecutivos con $\gamma_d - 1$ y 1 partidos sin cambiar el orden de los mismos, este nuevo fixture es óptimo para $RDP(n, p + 1)$.*

Demostración:
Luego de separar el período d en dos períodos consecutivos, digamos $\bar{d}$ y d' con $\gamma_d - 1$ y 1 partidos, respectivamente, se tendrá un nuevo valor RD, z'. Supongamos que hay un partido (i', j') en el período d' en la fecha $f - 1$, y que existen dos partidos (i', j) e (i, j') en la fecha f. Los equipos i y j jugaron en un período distinto a d' en la fecha $f - 1$ antes de la división, y los partidos (i', j) e (i, j') tenían alguna diferencia de descanso que era óptima. La única nueva diferencia en los descansos puede provenir de generar el período d'. Puesto que tanto i' como j', que juegan en d', deben enfrentar a equipos que vienen de otros períodos generando nuevas diferencias en los partidos (i', j) e (i, j'). Cada uno de estos partidos tendrá un incremento de a lo sumo 1 en la diferencia de descansos, resultando en un incremento de a lo sumo 2 en la fecha f. Así, la nueva solución óptima z' debe ser a lo sumo $LB + 2.(n - 2)$. Ya que esta división del período d agrega un período de un solo partido y p_1 es par ($p_1 + 1$ es impar), la nueva cota inferior resulta $LB' = (n - 2).[p_1 + 1 + mod(p_1 + 1, 2)] = (n - 2).[p_1 + 2] = LB + 2.(n - 2)$. Ya que el nuevo óptimo z' no puede ser menor que LB' (es decir, $z' \geq LB + 2.(n - 2)$), y las diferencias adicionales por la división son a lo sumo $2.(n - 2)$ (es decir, $z' \leq LB + 2.(n - 2)$), el nuevo fixture debe ser óptimo. □

Teorema 9. *Asumamos que tenemos un fixture óptimo para el problema $RDP(n, p)$ con $RD = LB$. Supongamos que existe un período d con $\gamma_d = 2$. Si el período d se divide en dos períodos consecutivos de un solo partido sin cambiar el orden de los partidos, el nuevo fixture se mantiene siendo óptimo para el problema $RDP(n, p+1)$.*

Demostración:
Si el óptimo RD antes de la división era 0, todos los equipos de cierta fecha f ($f > 1$) deben haber jugado en el mismo período en la fecha $f - 1$. Esto significa que todos los equipos que jugaron en el período d antes de la separación, también deben jugar entre sí a lo largo del torneo (si juegan 1vs2 y 3vs4 en el día d, 1 debe jugar contra 3 y 4). Por eso, cuando el período d se divide, los dos partidos entre estos equipos van a generar una diferencia de descanso de 2 por fecha, y el valor de RD va a aumentar $2.(n-2)$ en el torneo. Por el teorema 7, este valor es igual al LB para un fixture con $p_1 = 2$, el fixture es óptimo. En el caso en que $p_1 > 0$ y RD es positivo (no es 0 e igual al LB), todos los equipos que juegan en períodos de más de un partido deben haber jugado en el mismo período en la fecha anterior, ya que todas las diferencias de descanso existentes deben provenir de períodos de un solo partido. Cuando d se separa en dos períodos consecutivos de un partido, los partidos de esos nuevos períodos van a generar un adicional de $2.(n-2)$ diferencias aumentando el valor de RB a $LB + 2.(n-2)$. Este último coincide con el LB para el nuevo fixture con $p_1 + 2$ períodos de un partido, y por lo tanto, es óptimo. □

2.3.2. Método de tiempo polinomial para algunos casos especiales

El método que veremos a continuación también fue propuesto por los autores de [24]. Este último es de tiempo polinomial y puede ser aplicado al $RDP(n, p|\gamma_1, ..., \gamma_p)$ para $n = 2^k$ $(n \geq 8)$ donde $k \in \mathbb{Z}^+$, y el número de partidos por período es divisible por 2 $(mod(\gamma_d, 2) = 0 \quad \forall d = 1, ..., p)$. El fixture final dado por el método garantiza una diferencia total en los descansos igual a 0. Además, se obtienen fixtures óptimos para casos con períodos con un solo partido, cuando el número de estos períodos es par, debido a los teoremas 8 y 9.

Para construir el fixture óptimo, se deben atravesar tres pasos. En el primer paso, se separan los equipos en grupos de 4 cada uno distribuidos sobre $n/4$ días, los cuales contienen todas las posibles combinaciones de enfrentamientos entre ellos necesarias para un torneo Round-Robin. Los grupos siguientes se forman intercambiando dos equipos de dos períodos distintos entre sí. Luego, se muestra que la diferencia de descansos que ocurre al finalizar los intercambios puede ser eliminada. Segundo, se generan los partidos de cada grupo siguiendo el método del círculo. En el tercer paso, se eliminan las diferencias de descanso aplicando los intercambios necesarios entre los partidos en ciertas fechas.

Paso 1: Construir los grupos

Los grupos se construyen de la siguiente manera: comenzando con la primera fecha, se dividen los equipos en grupos de 4 equipos donde en cada uno los equipos tienen números consecutivos. De esta forma, se asigna el grupo (1,2,3,4) al primer período, el grupo (5,6,7,8) al segundo, etc., hasta alcanzar el período $n/4$ que incluye al grupo $(n-3, n-2, n-1, n)$. Una vez armado el fixture, se van a jugar todos

Agrupaciones	Período 1		Período 2		Período 3		Período 4	
a	1-2	3-4	5-6	7-8	9-10	11-12	13-14	15-16
b	1-2	5-6	3-4	7-8	9-10	13-14	11-12	15-16
c	1-2	7-8	3-4	5-6	9-10	15-16	11-12	13-14
d	1-2	9-10	3-4	11-12	7-8	13-14	5-6	15-16
e	1-2	11-12	3-4	9-10	5-6	15-16	7-8	13-14
f	1-2	13-14	3-4	15-16	5-6	9-10	7-8	11-12
g	1-2	15-16	3-4	13-14	5-6	11-12	7-8	9-10

Cuadro 2.10: Agrupación de equipos para n = 16 obtenido con intercambios

los partidos posibles del grupo antes de continuar con los partidos de los siguientes grupos. Para generar los siguientes grupos, se toman dos períodos diferentes del conjunto actual de grupos, y se intercambian dos equipos de cada período entre sí, de manera tal que todos los pares de equipos van a estar agrupados con otro par proveniente de un grupo diferente. Análogamente, el siguiente grupo de equipos es generado intercambiando los equipos del segundo conjunto formado. Cada vez que se produce un nuevo conjunto de grupos de equipos, va a haber $n/8$ intercambios. Se continúa generando nuevos conjuntos de grupos a través de intercambios de equipos hasta agotar todas las combinaciones posibles, que serían $n/2 - 1$. En la tabla 2.10 se puede ver este paso ejemplificado para $RDP(16, 4|2, 2, 2, 2)$, donde los pares de equipos con mismo color son los intercambiados. Notar que estas agrupaciones son solo combinaciones de 4 equipos y no los partidos jugados.

Paso 2: Generar los partidos

En este paso se generan los partidos para completar el fixture. Usando el método del círculo, se generan todos los partidos posibles de cada grupo manteniendo a los equipos siempre en el mismo día. En la primera fecha, donde los equipos todavía no se enfrentaron, se pueden generar 6 partidos de cada grupo. Considerando 2 partidos por período, quedan conformadas las primeras 3 fechas. Similarmente, en las fechas siguientes, luego de cada reagrupación de equipos, se juegan todos los partidos posibles respetando el método del círculo sin modificar los períodos. Notar que en todas las fechas, salvo la primera, dos partidos de cada grupo ya van a haberse jugado previamente. Los restantes 4 partidos de cada grupo pueden jugarse en dos fechas. En la tabla 2.11 vemos este paso para el ejemplo que se estaba analizando.

Paso 3: Eliminar las diferencias de descansos

Debido a los intercambios hechos al construir los grupos, los equipos equipos que juegan en períodos distintos en fechas impares (salvo la fecha 1) van a tener que enfrentarse en fechas pares (a partir de la 4) lo que va a generar diferencias en los descansos. Observar que se pueden intercambiar dos partidos jugados en dichas fechas impares involucrando a los equipos que se intercambian cuando se forman los nuevos grupos para deshacerse de las diferencias de descanso. Este intercambio de

Agrupaciones	Fecha	Período 1		Período 2		Período 3		Período 4	
a	1	1-3	2-4	5-7	6-8	9-11	10-12	13-15	14-16
	2	1-4	3-2	5-8	7-6	9-12	11-10	13-16	15-14
	3	1-2	3-4	5-6	8-7	9-10	12-11	13-14	16-15
b	4	1-5	2-6	3-7	4-8	9-13	10-14	11-15	12-16
	5	1-6	2-5	3-8	4-7	9-14	10-13	11-16	12-15
c	6	1-7	2-8	3-5	4-6	9-15	10-16	11-13	12-14
	7	1-8	2-7	3-6	4-5	9-16	10-15	11-14	12-13

Cuadro 2.11: Generando las primeras 7 fechas del $RDP(16, 4|2, 2, 2, 2)$

partidos no genera diferencias ya que ambos equipos jugaron en el mismo período la fecha anterior. Por ejemplo, en la tabla 2.11 uno puede reemplazar el partido 3-4 por el partido 5-6 en la fecha 3, sin agregar diferencias de descanso en esa fecha y eliminando las de ese período de la fecha 4. Por último, en la tabla 2.12 podemos ver como resulta el óptimo del problema $RDP(16, 4|2, 2, 2, 2)$ con este algoritmo.

Con los pasos anteriormente explicados, los autores probaron que el fixture resultante es óptimo con diferencia de descansos igual a 0 cuando tenemos esas condiciones.

Agrupaciones	Fecha	Período 1		Período 2		Período 3		Período 4	
a	1	1-3	2-4	5-7	6-8	9-11	10-12	13-15	14-16
	2	1-4	3-2	5-8	7-6	9-12	11-10	13-16	15-14
	3	1-2	5-6	3-4	8-7	9-10	13-14	12-11	16-15
b	4	1-5	2-6	3-7	4-8	9-13	10-14	11-15	12-16
	5	1-6	4-7	3-8	2-5	9-14	12-15	11-16	10-13
c	6	1-7	2-8	3-5	4-6	9-15	10-16	11-13	12-14
	7	1-8	9-16	3-6	11-14	2-7	10-15	4-5	12-13
d	8	1-9	2-10	3-11	4-12	5-13	6-14	7-15	8-16
	9	1-10	4-11	3-12	2-9	5-14	8-15	7-16	6-13
e	10	1-11	2-12	3-9	4-10	5-15	6-16	7-13	8-14
	11	1-12	8-13	3-10	6-15	5-16	4-9	7-14	2-11
f	12	1-13	2-14	3-15	4-16	5-9	6-10	7-11	8-12
	13	1-14	4-15	3-16	2-13	5-10	8-11	7-12	6-9
g	14	1-15	2-16	3-13	4-14	5-11	6-12	7-9	8-10
	15	1-16	2-15	3-14	4-13	5-12	6-11	7-10	8-9

Cuadro 2.12: Generando las primeras 7 fechas del $RDP(16, 4|2, 2, 2, 2)$

Teorema 10. *El resultado de aplicar los pasos Paso 1: Construir los grupos, Paso 2: Generar los partidos y Paso 3: Eliminar las diferencias de descansos a un conjunto de n equipos, donde n es un entero positivo potencia de 2 y $n \leq 8$ y el número de partidos por período es par, es un fixture con 0 diferencias en los descansos.*

2.3.3. Algoritmo de tiempo polinomial para Fixtures canónicos

Un fixture canónico es aquel construido utilizando el método del círculo. Es un fixture muy famoso que se utiliza en muchos torneos debido a su simplicidad para armarse. Por ejemplo, dos ligas importantes que utilizaban este tipo de fixture son la Premier League (Rusia) y la Süper Lig (Turquía). Actualmente, es el sistema elegido en la Liga Argentina para definir el fixture del torneo. La razón principal para utilizarlo es que es posible crear un fixture asignando localías con $n-2$ *breaks* en un torneo Round-Robin, el cual es la cota inferior teórica del problema de minimización de breaks [7].

En un fixture canónico, para n par, los partidos de cada fecha $f \in \{1, ..., n-1\}$ pueden jugarse de la siguiente forma:

$$Partidos_f = \{[n, f]\} \cup \{[f+j, f-j] : j = 1, ..., n/2 - 1\} \tag{2.1}$$

donde los partidos $r+j$ y $r-j$ están expresados como $r+j(mod\ n-1)$ y $r-j(mod\ n-1)$, respectivamente [7]. Cuando alguno de estos valores es 0, se reemplaza por $n-1$.

A continuación, vamos a ver un algoritmo de tiempo polinomial que resuelve el $RDP(n, k)$ para este tipo de fixtures. El mismo fue presentado en [5]. Supongamos que tenemos n (par) equipos con $n-1$ fechas y p períodos. Notamos P_f la cantidad de períodos de la fecha f. Primero, se asignan los partidos de cada fecha a índices de partidos desde 1 hasta $n/2$ usando el orden determinado por 2.1. Esto es, para cada fecha f, al partido $[n, f]$ le asignamos el índice 1 y para cada j entre 1 y $n/2-1$, al partido $[f+j, f-j]$ le asignamos el índice j+1. Luego, en cada fecha f, asignamos $nPartidos_1^f$ partidos con los índices más bajos al período 1, $nPartidos_2^f$ partidos con los siguientes índices más bajos al período 2, y así sucesivamente. Como resultado de estas asignaciones, los índices de los partidos son siempre consecutivos. Notemos que en este caso, la cantidad de partidos por período puede variar. En la tabla 2.13 podemos ver un ejemplo de un fixture canónico para $n = 14$ y $p = 3$. Observar que cada partido se asigna a un índice y, en cada fecha, los períodos están separados utilizando dos barras verticales ($\|$).

En algunos fixtures, puede desearse que no se juegue en algún período (es decir, $\exists k \in P_f$ tal que $nPartidos_k^f = 0$). De ahora en más, si un período tiene al menos un partido, se notará como período distinto de 0. Sea p'_f el número de períodos distintos de 0 de la fecha f y $P'_f = \{1, ..., p'_f\}$ su conjunto. Cada $k' \in P'_f$ está mapeado a un período $k \in P_f$ con $t_f(k') = k$ tal que el período k es el k'-ésimo período distinto de 0 de P_f.

Teorema 11. *Para fixtures canónicos, el algoritmo genera* $\sum_{f=1}^{n-2} 2.(t_f(p'_f) - t_f(1))$ *diferencias de descanso en todo el torneo, el cual es óptimo.*

Demostración:
Para probar la optimalidad de la solución, primero veamos cual es la solución del algoritmo para cada descomposición del problema. Luego, se observa que la cota

Fecha	Período 1 ‖	Período 2 ‖	Período 3 ‖				
1	1-14	2-13‖	3-12	4-11	5-10‖	6-9	7-8
2	2-14	3-1*	4-13*‖	5-12	6-11*‖	7-10*	8-9
3	3-14	4-2‖	5-1*	6-13*‖	7-12*	8-11*	9-10
4	4-14	5-3*	6-2*‖	7-1*	8-13*	9-12‖	10-11
5	5-14	6-4	7-3*	8-2*‖	9-1‖	10-13*	11-12*
6	6-14‖	7-5	8-4	9-3*	10-2**	11-1*‖	12-13
7	7-14*	8-6*‖	9-5	10-4	11-3	12-2*‖	13-1*
8	8-14	9-7*‖	10-6*	11-5	12-4‖	13-3*	1-2*
9	9-14	10-8*	11-7*‖	12-6‖	13-5*	1-4*	2-3
10	10-14	11-9	12-8*	13-7**	1-6*‖	2-5‖	3-4
11	11-14	12-10	13-9‖	1-8	2-7*‖	3-6**	4-5*
12	12-14	13-1‖	1-10*‖	2-9*	3-8*	4-7*	5-6
13	13-14‖	1-12*‖	2-11**	3-10*	4-9	5-8	6-7
índices	1	2	3	4	5	6	7

Cuadro 2.13: $RDP(14, 3)$

inferior a la diferencias de descansos de cada problema es igual al valor del algoritmo, y por lo tanto este es exacto. El valor de las diferencias totales de descanso del torneo se obtiene sumando las soluciones de cada problema.

Consideremos un partido entre dos equipos, digamos $[i, j]$, en la fecha $f + 1$. La diferencia en los descansos de este partido de los períodos en los que jugaron ambos equipos en la fecha f. Asumimos, sin perdida de generalidad, que el fixture canónico se construye como en el algoritmo, es decir, fijando el equipo n y rotando al resto en el sentido contrario de las agujas del reloj. Entonces, el índice de partido de cada equipo puede cambiar a lo sumo en 1. Por lo tanto, cuando se enfrentan los equipos i y j en la fecha $f + 1$, en la fecha f los separaba a lo sumo 2 índices en sus partidos, donde posiblemente había otro índice entre ellos. Si se asume que los períodos distintos de 0 son consecutivos (como suele suceder en la práctica), la diferencia resultante en el partido $[i, j]$ será de a lo sumo 2 si se aplica el algoritmo. Más aún, en ambos extremos del círculo, donde i y j jugaron en partidos con índices consecutivos, la máxima diferencia posible de descansos es 1. Así, la diferencia en los descansos de un partido en la fecha $f + 1$ puede ser 0, 1 o 2. En el caso general, cuando los períodos distinto de 0 no son consecutivos, la diferencia de descansos va a ser igual a la respectiva diferencia en los períodos de los partidos de los equipos en la fecha previa.

Veamos los posibles casos ejemplificados. En las siguientes figuras, las lineas punteadas notan la división entre dos períodos distintos de 0. Como se notó antes, los períodos distintos de cero se notan con un apóstrofe. Además, se pueden tener en cuenta únicamente estos períodos, ya que los que tienen cero partidos, afectan a todos por igual. En la figura 2.4 se pueden observar los tres posibles casos en los que no hay diferencias en los descansos de los equipos. El caso (a) es aquel que sucede cuando ambos equipos i y j juegan en el mismo período intermedio distinto

de cero (digamos k') en la fecha f. La figura 2.4 (b) muestra el caso en el que no hay diferencias en el extremo izquierdo del círculo, donde j es el equipo fijo. Por último, en (c), se observa que pasa en el extremo derecho del círculo, donde ambos pertenecen al último período. Puesto que en los tres casos los equipos juegan en el mismo período, la diferencia siempre es cero.

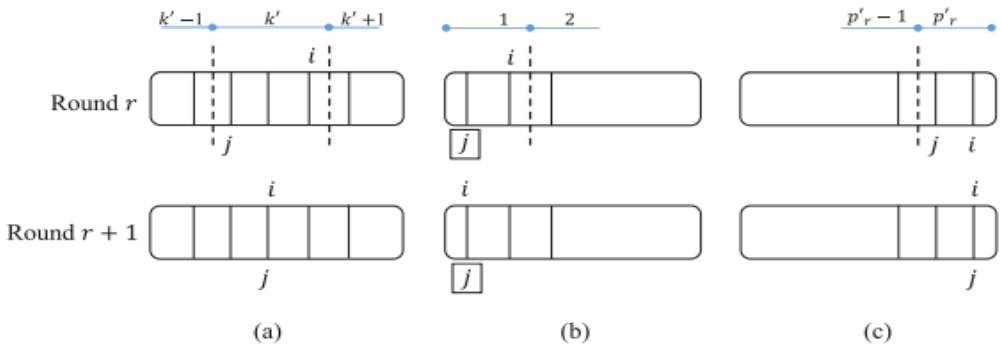


Figura 2.4: Caso en el que no hay diferencia en los descansos

La figura 2.5 muestra cuatro casos en donde la diferencia de descansos Δ, la cual es igual a la diferencia entre dos períodos distintos de cero. En 2.5 (a) el equipo i juega en el período $k'+1$ distinto de cero de la fecha f mientras que el equipo j juega en el k' ($\Delta = t_f(k'+1) - t_f(k')$). El caso (c) es muy similar al anterior, pero con i jugando en el período k' distinto de cero y j en el $k'-1$, ambos en la fecha f ($\Delta = t_f(k') - t_f(k'-1)$). La imagen 2.5 (b) ilustra lo que sucede en los primeros dos períodos distintos de cero al ser j el equipo fijo ($\Delta = t_f(2) - t_f(1)$). Por último, en (d) se tiene el caso del período final, en el extremo derecho del círculo ($\Delta = t_f(p'_f) - t_f(p'_f - 1)$).

Finalmente, la figura 2.6 muestra el caso en el que la diferencia de descansos es igual a la diferencia que hay entre dos períodos distintos de cero que difieren en un período distinto de cero entre sí. En el ejemplo, el equipo i juega en la fecha f en el $k'+1$-ésimo período distinto de cero y el j en el $k'-1$-ésimo. Cuando se enfrentan en

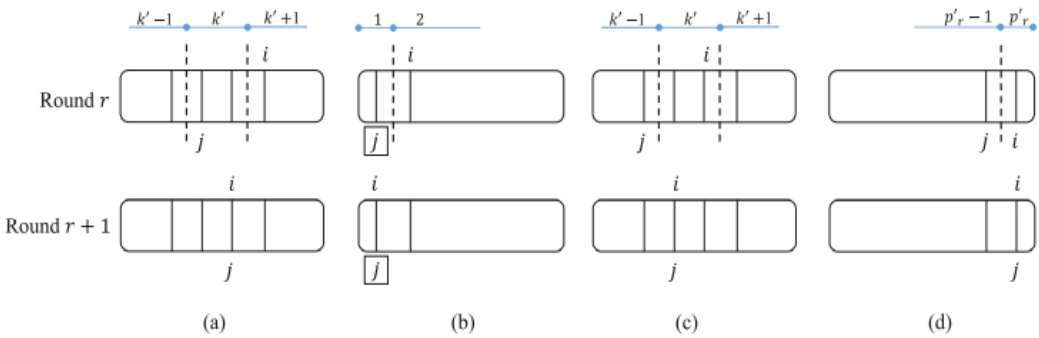


Figura 2.5: Caso en el que hay un período de diferencia en los descansos

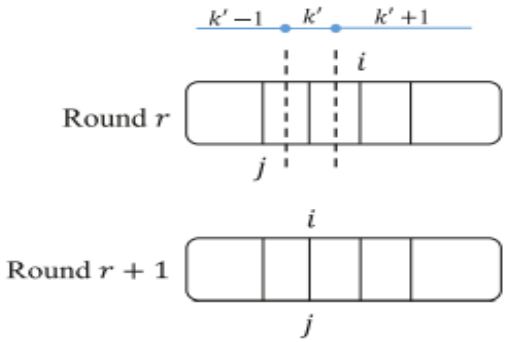


Figura 2.6: Caso en el que hay dos períodos de diferencia en los descansos

la fecha $f+1$, la diferencia de descansos va a ser igual a la diferencia de períodos entre los períodos distintos de cero, digamos $\Delta = \Delta_1 = \Delta_2$ donde $\Delta_1 = t_f(k'+1) - t_f(k')$ y $\Delta_2 = t_f(k') - t_f(k'-1)$.

Notar que cada linea punteada es cruzada por dos equipos al hacer cada rotación: uno en la fila superior del lado derecho de la linea y otro en la fila inferior a la izquierda de la linea. Además, cada linea separa dos períodos distinto de cero, por lo que entre cada uno de estos pueden existir períodos sin partidos. Cada una de estas contribuye a la diferencia de descansos de dos partidos distintos. Más aún, la diferencia de descansos de un partido viene dada por la suma de las diferencias de los períodos separados por las lineas punteadas que ambos equipos cruzaron, si existen. Por eso, la diferencia total de descansos de la fecha $f+1$ es igual a la suma de las diferencias de períodos generadas por los equipos que cruzaron una linea punteada en la fecha anterior, dado por $2.\sum_{k'=2}^{p'_f}(t_f(k') - t_f(k'-1))$. Esta expresión se reduce a $2.(t_f(p'_f) - t_f(1))$. De este modo, si hay p'_f períodos distintos de cero en una fecha $f \in \{1, ..., n-2\}$, el algoritmo propuesto para fixtures canónicos genera una diferencia en los descansos de $2.(t_f(p'_f) - t_f(1))$ en la fecha $f+1$. Así, el número total de diferencias de descanso depende únicamente de $t_f(p'_f)$ y $t_f(1)$, es decir, el primer y último período distinto de cero de cada fecha.

El siguiente paso es encontrar una cota inferior a las diferencias en los descansos para un subproblema. Para eso, primero hacemos algunas observaciones generales sobre diferencias en los descansos cuando se aplica el método del círculo. Consideremos todos los partidos indexados de 1 a $n/2$ para cualquier fecha. Según el método, dos equipos de los partidos 1 y 2, y dos equipos de los partidos $n/2-1$ y $n/2$ juegan entre sí en la siguiente ronda. Para el resto de los partidos, sucede lo siguiente: los equipos que juegan en partidos con índices impares enfrentan en la siguiente fecha a equipos que también jugaron en partidos con índices impares, y lo mismo sucede con los pares. Por lo tanto, para que no exista diferencia en los descansos, todos los partidos impares y pares deben pertenecer al mismo período. Cuando $p'_f = 1$, el total de diferencias en la fecha $f+1$ es $2.(t_f(1) - t_f(1)) = 0$, ya que todos pertenecen al mismo período.

Veamos una prueba por inducción en p'_f cuando se agrega un nuevo período

distinto de cero a un conjunto de períodos distintos de cero ya existentes. Asumamos, sin perdida de generalidad, que el tiempo del nuevo período k' es siempre posterior a los que existen (es decir, $t_f(k') \quad > \quad t_f(k'-1) \quad \forall k' = 2, ..., p'_f$). Consideremos el caso $p'_f = 1$, donde se agrega un período distinto de cero luego del existente, al cual se le deben asignar algunos partidos. Para no tener diferencias en los descansos de la siguiente fecha, asignamos todos los partidos con índices impares a este segundo período (también podrían asignarse los pares y aplicar argumentos similares). Puesto que los partidos de los extremos del círculo se jugaron en períodos distintos, y dos equipos de cada extremo se enfrentan en la siguiente fecha, existe una diferencia en los descansos de $\Delta = 2.(t_f(2) - t_f(1))$. Para deshacerse de estas diferencias, debemos asignar los partidos 1, 2, $n/2-1$ y $n/2$ al segundo período distinto de cero (también podrían agregarse al primero, o un par de partidos para cada período y los argumentos utilizados serían similares). Pero esto fuerza a algunos equipos de los partidos pares a jugar contra equipos de estos pares de partidos del segundo período, lo cual sigue generando una diferencia de $\Delta = 2.(t_f(2) - t_f(1))$. Si continuamos asignando partidos pares al segundo período distinto de cero con el fin de deshacernos de estas nuevas diferencias, algunos partidos deben mantenerse en el primer período, ya que de lo contrario tendríamos un solo período distinto de cero. Incluso si hay un único partido asignado al primer período, va a resultar en la siguiente fecha en la diferencia de descansos antes dicha. Ya que posiblemente haya más de un partido asignado al primer período, la diferencia de descansos será como mínimo $\Delta = 2.(t_f(2) - t_f(1))$, lo cual concluye el case base.

Asumamos que $p'_f = k'$ y por hipótesis inductiva el total de diferencias en los descansos en la siguiente fecha es al menos $\Delta = 2.(t_f(k') - t_f(1))$ para algún $k' > 2$. Queremos ver que luego de asignar algunos partidos al período $k'+1$ distinto de cero (es decir, p'_f se convierte en $k'+1$), el total de las diferencias debe ser al menos $\Delta = 2.(t_f(k'+1) - t_f(1))$. Para tener un período $k'+1$ distinto de cero, debemos elegir algunos partidos (algún partido) que previamente fueron asignados a otros períodos y cambiarlos al período $t_f(k'+1)$. Los partidos que no fueron asignados al nuevo período pueden o no mantener el período al que fueron asignado antes de la introducción del nuevo. Más allá de que suceda, estos partidos pertenecen a los períodos de $t_f(1)$ a $t_f(k')$. Si los partidos intercambiados generan diferencias en los descansos, la diferencia total va a aumentar en al menos $2.(t_f(k'+1) - t_f(k'))$, ya que $t_f(k'+1)$ está al menos a $(t_f(k'+1) - t_f(k'))$ distancia de los períodos que actualmente existen. Por otro lado, si estos partidos elegidos causan nuevas diferencias, incluso estando todas estas entre los períodos k' y $k'+1$, entonces la diferencia de descansos por la introducción va a incrementarse en al menos $2.(t_f(k'+1) - t_f(k'))$, ya que agregar un nuevo período genera al menos 2 diferencias más. Por eso, luego de la introducción del $k'+1$-ésimo período distinto de cero, el aumento de diferencias en la fecha $f+1$ será de al menos $2.(t_f(k'+1) - t_f(k'))$. Una vez que todos los partidos fueron asignados a períodos distintos de cero hasta el $k'+1$, la diferencia total de descansos en la fecha $f+1$ es al menos $2.(t_f(k') - t_f(1)) + 2.(t_f(k'+1) - t_f(k')) = 2.(t_f(k'+1) - t_f(1))$. Por inducción, las diferencias totales de descansos en la fecha $f+1$ con p'_f períodos distintos de cero en la fecha f es al menos $2.(t_f(p'_f) - t_f(1))$.

Como la cota inferior $2.(t_f(p'_f)-t_f(1))$ es igual al valor dado por el algoritmo, este es exacto. Más aún, ya que las diferencias en los descansos ocurren en la asignación de partidos a períodos en todas las fechas salvo una, el valor óptimo del total de diferencias en los descansos en el torneo entero es igual a $\sum_{f=1}^{n-2} 2.(t_f(p'_f)-t_f(1))$ $\square$

Observar que cuando los períodos distinto de cero son consecutivos, el óptimo de cada fecha pasa a ser $2.(p'_f - 1)$ ya que $t_f(p'_f) - t_f(1) = p'_f - 1$. En la tabla 2.13, los períodos son consecutivos y $t_f(p'_f) = 3$ para todas las fechas. Los partidos con una estrella denotan una diferencia de un período, mientras que los que tienen dos estrellas difieren de dos períodos con respecto a la fecha anterior.

Capítulo 3

Liga Profesional de Fútbol Argentina

La Asociación de Fútbol Argentino (AFA) es el organismo encargado de organizar y regular las distintas competiciones relacionadas a fútbol que hay en nuestro país. Desde el año 2020, se creó la Liga Profesional de Fútbol (LPF), un organismo interno de la AFA que se dedica exclusivamente a organizar el torneo de la primera división del fútbol argentino.

Actualmente, en la primera división del fútbol argentino participan 28 equipos. Luego de dos temporadas sin descensos debido a la Pandemia, se retomarán a fin de año. En los últimos años, el fútbol argentino sufrió diversas modificaciones en lo que al torneo respecta. Pasó por todo tipo de torneo, variando constantemente el número de equipos participantes de acuerdo a los intereses de los dirigentes de AFA de turno.

3.1. Formato

Hoy en día, la liga profesional de fútbol consta de dos torneos con formatos distintos:

- **Single Round Robin por grupos + Playoffs**
- **Single Round Robin**

El primero de estos torneos, conocido como la Copa de la Liga Profesional, consta de dos grupos (cuyo tamaño depende de la cantidad total de equipos en la primera división) en los cuales se juega un Single Round Robin. Los mejores 4 posicionados en la tabla de cada zona clasifican a una etapa de Playoff. Los enfrentamientos se dan de la siguiente forma: el 1º de cada zona contra el 4º de la otra zona y los 2º contra los 3º enfrentando a aquellos que pertenecen a distintas zonas. Estos enfrentamientos se juegan en los estadios de los equipos mejor posicionados en su respectiva tabla, y a partir de semifinales, se juegan en una cancha neutral.

La competencia que se realiza la segunda mitad del año, es un Single Round Robin de 28 equipos (26 en 2021).

Al final del año, los ganadores de ambas competencias se enfrentan en una única final llamada Copa de Campeones. Además, a lo largo del año de computa una tabla paralela con los puntos que acumula cada equipo en ambas competencias. Dicha tabla tiene como objetivo tanto determinar los equipos que clasifican a las copas internacionales del próximo año junto a los campeones, como también participar en el cálculo de los promedios para definir los descensos a la segunda división.

3.2. Motivación

Este trabajo está enfocado en el torneo del segundo semestre. Usualmente, las ligas tienen alrededor de 20 equipos en primera división y se juegan en formato doble Round Robin. Las temporadas comienzan en Agosto y finalizan en Julio, con un corte de dos semanas durante las fiestas. En consecuencia, se juegan 19 partidos de la liga en cada mitad de año. A estos partidos se le agregan los torneos internacionales y las copas nacionales, los cuales suelen jugarse en los días de semana.

Por estas razones, los equipos que participan en todas las competiciones, suelen tener calendarios más apretados con menos descanso. Surge entonces la necesidad de equiparar esto minimizando la diferencia en los descansos de los equipos.

Cabe aclarar que en el año actual hay Mundial, mes en el cual toda competición de fútbol entra en un parate. Además, a diferencia del resto de los mundiales, este año se juega en Noviembre, cortando el segundo semestre. Todo esto hace que el calendario sea mucho mas apretado para las ligas y las copas.

3.3. Modelo

El modelo toma como input un Fixture completo y asigna los partidos de cada fecha a un determinado día. Dicho Fixture, es el armado por la Liga Profesional de Fútbol, que se juega durante el segundo semestre del año. Pueden verse las fechas programadas en la página de la organización [27]. El modelo aquí planteado es una modificación del presentado en [9] (con sugerencias de [19]).

Conjuntos y parámetros

$Equipos$: Conjunto de equipos.

$Fechas$: Conjunto de fechas.

$Partidos$: Conjunto de tuplas (i, j, k) dadas por el Fixture tal que el equipo i

juega de local contra el equipo j en la fecha k ($Part$ en el modelo).

$Dias_k$: Conjunto de días del torneo en los cuales se juega la fecha k.

$Fijos$: Conjunto de partidos que deben jugarse en un día específico.

$Equipos_{Pop} \subseteq Equipos$: Conjunto de equipos populares.

$Copas$: Conjunto de tuplas (i, k, d) tal que i no puede jugar el día d de la fecha k para cumplir con los días requeridos de descanso debido a otro torneo.

$u_{i,j}$: Máxima diferencia entre los días de descanso del equipo i y el equipo j antes de su partido.

$MaxPartidos_d$: Máximo número de partidos que se pueden jugar el día d.

$MaxPop_d$: Máximo número de partidos que pueden ser jugados por equipos populares el día d.

Variables de decisión

Para cada $(i, j, k) \in Partidos$ y para cada $d \in Dias_k$ se definen las variables binarias $y_{i,j,k,d}$:

$$y_{i,j,k,d} = \begin{cases} 1 & \textit{si el equipo } \boldsymbol{i} \textit{ juega de local contra el equipo } \boldsymbol{j} \textit{ en la fecha } \boldsymbol{k} \textit{ el día } \boldsymbol{d} \\ 0 & \textit{si no} \end{cases}$$

Por otro lado, para cada $(i, j, k) \in Partidos$, con $k > 1$, para cada $d \in Dias_{k-1}$ y $\hat{d} \in Dias_{k-1}$ se tienen las variables binarias $x_{i,j,k,d,\hat{d}}$:

$$x_{i,j,k,d,\hat{d}} = \begin{cases} 1 & \textit{si el equipo } \boldsymbol{i} \textit{ juega el día } \boldsymbol{d} \textit{ en la fecha } \boldsymbol{k-1} \textit{ y el equipo } \boldsymbol{j} \\ & \textit{juega el día } \boldsymbol{\hat{d}} \textit{ en la fecha } \boldsymbol{k-1} \\ 0 & \textit{si no} \end{cases}$$

Función objetivo

La función objetivo minimiza, sobre todos los partidos, la suma de todas las diferencias que hay en los descansos, entre dos equipos. lo cual es simplemente representado por la diferencia en los días que jugaron la fecha anterior:

$$min \sum_{\substack{(i,j,k) \in Part \\ k>1}} \sum_{d \in Dias_{k-1}} \sum_{\hat{d} \in Dias_{k-1}} |d - \hat{d}| x_{i,j,k,d,\hat{d}}$$

Restricciones

1. Asegura que las diferencias de los descansos entre dos equipos antes de su enfrentamiento no supere la cota dada.

$$|d-\hat{d}|x_{i,j,k,d,\hat{d}} \leq u_{i,j} \qquad \forall (i,j,k) \in Part, k > 1, \forall d \in Dias_{k-1}, \forall \hat{d} \in Dias_{k-1}$$

2. Relación lógica entre las variables.

$$\sum_{\substack{l \in Equipos: \\ (i,l,k-1) \in Part}} y_{i,l,k-1,d} + \sum_{\substack{l \in Equipos: \\ (l,i,k-1) \in Part}} y_{l,i,k-1,d} \geq x_{i,j,k,d,\hat{d}}$$

$$\forall (i,j,k) \in Part, k > 1, \forall d \in Dias_{k-1}, \forall \hat{d} \in Dias_{k-1}$$

$$\sum_{\substack{l \in Equipos: \\ (j,l,k-1) \in Part}} y_{j,l,k-1,\hat{d}} + \sum_{\substack{l \in Equipos: \\ (l,j,k-1) \in Part}} y_{l,j,k-1,\hat{d}} \geq x_{i,j,k,d,\hat{d}}$$

$$\forall (i,j,k) \in Part, k > 1, \forall d \in Dias_{k-1}, \forall \hat{d} \in Dias_{k-1}$$

$$\sum_{\substack{l \in Equipos: \\ (i,l,k-1) \in P}} y_{i,l,k-1,d} + \sum_{\substack{l \in Equipos: \\ (l,i,k-1) \in P}} y_{l,i,k-1,d} + \sum_{\substack{l \in Equipos: \\ (j,l,k-1) \in P}} y_{j,l,k-1,\hat{d}} + \sum_{\substack{l \in Equipos: \\ (l,j,k-1) \in P}} y_{l,j,k-1,\hat{d}} \leq x_{i,j,k,d,\hat{d}} + 1$$

$$\forall (i,j,k) \in Part, k > 1, \forall d \in Dias_{k-1}, \forall \hat{d} \in Dias_{k-1}$$

$Nota$: $P = Partidos$

3. Restricción que prohíbe a los equipos jugar dos partidos en días muy cercanos en fechas consecutivas, respetando el mínimo de 3 días de descanso.

$$\sum_{\substack{l \in Equipos: \\ (i,l,k-1) \in P}} y_{i,l,k-1,\hat{d}} + \sum_{\substack{l \in Equipos: \\ (l,i,k-1) \in P}} y_{l,i,k-1,\hat{d}} + \sum_{\substack{j \in Equipos: \\ (i,j,k) \in P}} y_{i,j,k,d} + \sum_{\substack{j \in Equipos: \\ (j,i,k) \in P}} y_{j,i,k,d} <= 1$$

$$\forall i \in Equipos, \forall k \in Fechas : k > 1, \forall d \in Dias_k, \forall \hat{d} \in Dias_{k-1} \quad si \quad d - \hat{d} < 3$$

4. Impone un máximo de partidos que pueden ser jugados por día.

$$\sum_{i \in Equipos} \sum_{\substack{j \in Equipos: \\ (i,j,k) \in Part}} y_{i,j,k,,d} \leq MaxPartidos_d \qquad \forall k \in Fechas,\ d \in Dias_k.$$

5. Limita la cantidad de equipos populares que pueden jugar en cada día.

$$\sum_{i \in Equipos_{Pop}} \sum_{\substack{j \in Equipos: \\ (i,j,k) \in Part}} y_{i,j,k,d} + \sum_{i \in Equipos_{Pop}} \sum_{\substack{j \in Equipos: \\ (j,i,k) \in Part}} y_{j,i,k,d} \leq MaxPop_d$$
$$\forall k \in Fechas,\ d \in Dias_k.$$

6. Garantiza que los equipos con otros torneos tengan al menos 3 días de descanso antes o después de su próximo partido.

$$\sum_{\substack{j \in Equipos: \\ (i,j,k) \in Part}} y_{i,j,k,d} + \sum_{\substack{j \in Equipos: \\ (j,i,k) \in Part}} y_{j,i,k,d} = 0 \qquad \forall (i,k,d) \in Copas$$

7. Partidos importantes que se fijan cierto día por requerimientos televisivos.

$$y_{i,j,k,d} = 1 \qquad \forall (i,j,k,d) \in Fijos.$$

8. Los partidos que pertenecen a cierta fecha deben jugarse en un día posible de esa fecha.

$$\sum_{d \in Dias_k} y_{i,j,k,d} = 1 \qquad \forall (i,j,k) \in Partidos.$$

9. Todos los días de cada fecha deben tener al menos un partido.

$$\sum_{i \in Equipos} \sum_{\substack{j \in Equipos: \\ (i,j,k) \in Part}} y_{i,j,k,,d} \geq 1 \qquad \forall k \in Fechas,\ d \in Dias_k.$$

10. Naturaleza de las variables.

$$y_{i,j,k,d} \in \{0,1\} \qquad \forall (i,j,k) \in Part, k > 1, \forall d \in Dias_{k-1}, \forall \hat{d} \in Dias_{k-1}$$

$$x_{i,j,k,d,\hat{d}} \in \{0,1\} \qquad \forall (i,j,k) \in Part, k > 1, \forall d \in Dias_{k-1}, \forall \hat{d} \in Dias_{k-1}$$

Capítulo 4

Resultados

En el presente capítulo veremos los distintos resultados obtenidos por el modelo aplicados a los torneos de la primera división del fútbol argentino de los años 2021 y 2022. Una variante para el trabajo, es considerar un paso inicial en el cual se crea el fixture del torneo. Este puede realizarse bajo alguno de los criterios mencionados anteriormente como minimizar los *breaks* o minimizar la distancia viajada por los equipos. Esto implicaría crear un modelo de programación lineal mixta en el cual, además del objetivo a minimizar, deberíamos incluir restricciones que los organizadores del torneo exijan. La desventaja de esto, es que no tenemos el acceso o contacto para saber estas restricciones, razón por la cual elegí utilizar el fixture dado por el mismo torneo. De esta manera, lo único que se compara son los días en los que se juegan los partidos y cuanto es el descanso en cada caso.

Los torneos en cuestión son single Round-Robin de 28 equipos para el año 2022 y 26 equipos el de 2021. Es decir, 27 y 25 fechas, respectivamente. Para poder hacer una comparación más exacta con el torneo original, se proveen además los días en los cuales deben jugarse cada una de las fechas. Por otro lado, el modelo tiene muchos parámetros que podemos manipular y observar como afecta esto en el resultado final. Toda esta información fue tomada de la página oficial de la Liga Profesional de Fútbol https://www.ligaprofesional.ar

4.1. Temporada 2021

El torneo del año 2021 consta de un single Round Robin de 26 equipos. Comencemos analizando los puntos básicos de paridad que debe tener un fixture de este tipo. Puesto que el torneo se desarrolla a lo largo de 25 fechas y no es un torneo a tiempo relajado, todos deben jugar un partido por fecha. Podemos observar entonces, que todos los equipos tiene entre 12 y 13 partidos de local y de visitante.

Otro aspecto que podemos analizar a simple vista es la cantidad de *breaks* que existen en el torneo y cuantos tiene cada equipo. Por empezar, en todo torneo Round Robin, la cantidad de equipos con cero *breaks* en todo el torneo puede ser a lo sumo dos. Esto es fácil de ver fijando a dos equipos con cero *breaks* (deben intercalar un partido de local y otro de visitante todo el torneo, pero uno empezando en casa y el

otro fuera), e intentando hacer lo mismo con un tercer equipo. El tercero en discordia deberá jugar con los primeros dos a lo largo del torneo, por lo que necesariamente debe tener un *break*. Dicho esto, en el fixture propuesto por los organizadores hay dos equipos con cero *breaks*. En cuanto al resto de los equipos, todos tienen un único *break*, local para algunos y visitante para otros. De esta manera, podemos decir que en este aspecto el torneo es parejo.

Veamos ahora lo que nos interesa para el trabajo. En las tablas 4.1 y 4.2 podemos ver los descansos que hubo en el torneo de 2021. Las celdas marcadas de rosa son los *breaks* de local, mientras que las celestes indican los de visitantes.

Una restricción usual que suele darse en los modelos en los cuales se busca minimizar los *breaks* totales del torneo, es pedir que ningún equipo tenga un *break* en las primeras dos ni últimas dos fechas, debido a que son momentos importantes del campeonato. En este aspecto, el torneo no es del todo parejo debido a que River comienza con dos partidos de local, mientras que Boca arranca con dos partidos fuera de casa.

En la tabla 4.2 tenemos además el mínimo y máximo número de días que descansó cada equipo en el torneo, como así el promedio a lo largo del campeonato. Observemos que todos los equipos descansan al menos tres días en cada una de las fechas, lo cual es un requerimiento importante que se debe cumplir. Más aún, solo dos equipos descansan siempre más de 3 días. Por otro lado, podemos ver que los promedios son sumamente parejos entre todos los equipos, variando en menos de un día entre todos ellos.

Fecha	Fecha 1	Fecha 2	Fecha 3	Fecha 4	Fecha 5	Fecha 6	Fecha 7	Fecha 8	Fecha 9	Fecha 10	Fecha 11	Fecha 12	Fecha 13	Fecha 14
Aldosivi	0	6	4	6	7	5	6	6	3	5	10	5	7	6
Argentinos	0	8	3	4	6	6	8	4	4	7	9	4	8	6
Arsenal	0	8	3	4	7	6	6	4	4	6	10	5	6	7
Atl. Tucumán	0	6	5	5	6	6	6	4	6	5	10	4	7	7
Banfield	0	5	4	4	6	7	6	4	4	8	10	5	5	9
Boca	0	8	3	5	7	7	6	4	4	6	10	4	8	7
C. Córdoba	0	5	5	4	7	6	6	3	5	5	11	5	6	6
Colon	0	6	3	4	8	5	7	4	4	7	10	5	7	8
DyJ	0	7	3	4	8	7	4	5	4	5	11	6	6	8
Estudiantes	0	9	4	4	7	6	5	6	4	7	9	4	5	9
Gimnasia	0	7	5	3	7	5	9	3	3	8	8	7	5	8
Godoy Cruz	0	6	3	3	7	9	5	5	3	7	8	5	8	6
Huracan	0	5	4	5	6	6	8	4	3	6	10	6	8	5
Independiente	0	7	3	3	8	6	6	4	4	8	8	6	7	6
Lanus	0	7	4	3	7	9	5	4	4	6	9	7	5	6
Newells	0	9	3	4	4	9	7	3	5	5	11	5	7	5
Patronato	0	8	3	5	7	6	6	3	6	6	10	4	7	6
Platense	0	6	5	3	6	9	6	4	3	7	10	5	5	8
Racing	0	7	3	4	8	7	6	3	5	7	8	5	9	6
River	0	7	3	4	6	7	8	4	4	6	10	4	6	8
R. Central	0	6	4	4	4	8	8	3	4	6	11	5	5	8
San Lorenzo	0	6	3	5	6	9	6	4	4	5	9	7	6	6
Sarmiento	0	9	3	3	6	10	4	4	6	5	9	5	6	9
Talleres	0	9	3	3	6	9	6	4	4	6	9	4	7	9
Union	0	9	3	3	7	6	9	4	4	7	9	5	6	5
Velez	0	8	3	5	6	6	7	4	4	8	7	5	6	8

Cuadro 4.1: LPF 2021 - Días de descanso - Fechas 1 a 14

Fecha	Fecha 15	Fecha 16	Fecha 17	Fecha 18	Fecha 19	Fecha 20	Fecha 21	Fecha 22	Fecha 23	Fecha 24	Fecha 25	Mínimo	Máximo	Promedio
Aldosivi	8	8	4	4	7	7	10	5	4	9	4	3	10	5,84
Argentinos	6	8	4	4	4	10	11	4	5	8	5	3	11	5,84
Arsenal	9	4	5	5	5	7	15	4	4	5	9	3	15	5,92
Atl. Tucuman	9	6	3	5	5	7	14	4	5	4	8	3	14	5,88
Banfield	5	7	4	4	6	8	11	6	6	6	7	4	11	5,88
Boca	6	7	4	4	6	9	12	4	6	4	7	3	12	5,92
C. Cordoba	7	7	4	4	9	6	13	4	5	4	8	3	13	5,8
Colon	7	6	4	3	8	5	15	4	3	5	8	3	15	5,84
DyJ	5	6	4	4	6	7	13	6	4	7	7	3	13	5,88
Estudiantes	5	7	5	4	7	5	14	4	4	8	5	4	14	5,88
Gimnasia	8	5	4	3	7	8	12	4	5	7	6	3	12	5,88
Godoy Cruz	9	4	5	4	7	6	13	4	4	7	7	3	13	5,8
Huracan	6	8	4	4	6	9	11	4	6	5	7	3	11	5,84
Independiente	9	6	4	4	6	6	14	4	6	5	7	3	14	5,88
Lanus	8	7	4	5	4	7	13	7	4	5	8	3	13	5,92
Newells	6	8	5	4	6	8	13	3	6	6	6	3	13	5,92
Patronato	7	6	4	6	7	6	11	5	4	6	8	3	11	5,88
Platense	6	7	4	4	7	7	15	4	4	5	6	3	15	5,84
Racing	6	6	4	4	6	8	15	4	4	5	7	3	15	5,88
River	6	8	4	4	6	7	14	4	3	7	6	3	14	5,84
R. Central	5	7	4	4	8	5	15	5	3	6	7	3	15	5,8
San Lorenzo	9	6	3	4	7	7	12	4	7	5	7	3	12	5,88
Sarmiento	6	7	5	4	5	8	13	4	6	4	7	3	13	5,92
Talleres	7	6	4	4	5	7	13	4	4	7	8	3	13	5,92
Union	7	8	3	6	6	8	10	7	4	4	8	3	10	5,92
Velez	6	7	4	5	6	8	12	4	4	6	7	3	12	5,84

Cuadro 4.2: LPF 2021 - Días de descanso - Fechas 15 a 25

Basados en esta tabla, podemos calcular las diferencias de descanso que se produjeron en cada fecha, para cada equipo, y por lo tanto, la diferencia total del torneo. Para esto, notamos con cero cuando un equipo no sufre diferencias en los descansos con respecto a su rival, y cuando existe diferencia, le asignaremos el valor de la diferencia al equipo que más descansa y el mismo valor negativo al que padece la desventaja. Esto puede verse en las tablas 4.3 y 4.4.

En las tablas anteriores podemos ver como fueron las diferencias en los descansos durante el torneo para cada equipo. Coloreamos con verde los partidos en los que los equipos tienen un día de ventaja con respecto a su rival, con azul a aquellos que descansan dos días más que su rival y con rojo a los pocos partidos que difieren de 3 días. Particularmente se observa cual fue la mejor ventaja que sacaron en estas diferencias como también la peor que sufrieron. Notemos que el máximo valor de diferencias es tres días de ventaja. La última columna nos indica la diferencia total de descansos que tuvo cada equipo en todo el torneo. La suma de todos estos valores es equivalente a la función objetivo del modelo propuesto. En total, el torneo tuvo 304 diferencias en los descansos.

Estamos en condiciones de comenzar a presentar los resultados obtenidos con el modelo. El modelo corrido consta de los siguientes parámetros: consideramos al conjunto de equipos populares $Equipos_{Pop}$ como los cinco equipos considerados grandes: Boca, River, San Lorenzo, Independiente y Racing. Debido a que en el fixture original todos los equipos tienen al menos tres días de descanso, agregamos la restricción de que los días de descanso para cada equipo sea al menos 3.

Fecha	Fecha 1	Fecha 2	Fecha 3	Fecha 4	Fecha 5	Fecha 6	Fecha 7	Fecha 8	Fecha 9	Fecha 10	Fecha 11	Fecha 12	Fecha 13	Fecha 14
Aldosivi	0	0	1	2	-1	-1	-1	2	-1	-1	2	0	2	1
Argentinos	0	-1	0	0	-1	-1	2	0	0	0	-1	-1	-1	-2
Arsenal	0	-1	0	0	0	-3	0	1	0	1	0	1	1	2
Atl. Tucumán	0	1	2	0	-1	1	2	0	1	-1	-1	0	2	1
Banfield	0	-3	1	-1	0	1	1	1	1	1	-1	-1	-2	1
Boca	0	3	0	2	1	1	0	0	-1	0	-1	0	1	-1
C. Córdoba	0	-1	2	0	0	0	0	0	1	0	1	0	0	0
Colon	0	-1	-1	1	2	0	1	0	0	2	0	0	-1	-1
DyJ	0	1	0	1	1	-3	-2	2	0	0	1	1	0	-1
Estudiantes	0	2	1	0	0	-1	-1	2	0	0	0	0	0	3
Gimnasia	0	0	1	-1	1	0	1	-1	0	1	1	2	-2	-1
Godoy Cruz	0	-1	-2	-1	1	0	-3	1	-1	-1	-2	0	1	1
Huracan	0	-1	1	1	-1	0	3	0	0	1	1	0	1	-2
Independiente	0	-2	0	0	0	-2	2	0	0	2	-1	0	-1	-2
Lanus	0	1	1	0	1	0	-2	1	1	1	1	2	-1	0
Newells	0	1	-1	-1	-2	2	-1	-2	-1	-2	1	-2	-1	-1
Patronato	0	-1	0	1	0	0	0	-1	2	0	1	0	1	-2
Platense	0	0	2	0	2	3	0	0	-1	2	1	1	0	2
Racing	0	0	-1	1	0	-2	0	0	1	-1	-1	1	1	-3
River	0	-2	-1	-1	-1	1	-1	-2	-2	-2	-1	-1	0	1
R. Central	0	-2	-1	-2	-2	2	1	-1	-1	0	1	-2	-2	2
San Lorenzo	0	1	0	1	-1	0	-2	-2	-2	-2	1	2	0	-1
Sarmiento	0	1	-2	-1	2	3	-2	0	2	-1	-1	0	0	1
Talleres	0	1	-2	-2	0	0	-3	0	0	0	-1	-1	2	1
Union	0	2	-1	0	1	0	3	-1	0	-1	0	-2	-1	-1
Velez	0	2	0	0	-2	-1	2	0	1	1	-1	0	0	2

Cuadro 4.3: LPF 2021 - Diferencias en los descansos - Fechas 1 a 14

Fecha	Fecha 15	Fecha 16	Fecha 17	Fecha 18	Fecha 19	Fecha 20	Fecha 21	Fecha 22	Fecha 23	Fecha 24	Fecha 25	Mínimo	Máximo	Promedio	Descansos totales
Aldosivi	2	2	-1	-1	-2	-2	-1	1	0	1	-1	-2	2	0,12	16
Argentinos	1	2	0	0	0	1	-2	0	0	-1	-2	-2	2	-0,28	6
Arsenal	0	-3	1	1	0	1	2	-3	-1	1	2	-3	2	0,12	14
Atl. Tucuman	2	-2	-2	-1	-2	-1	-1	-2	1	-1	2	-2	2	0	15
Banfield	-1	0	0	0	0	0	1	2	0	0	-2	-3	2	-0,04	10
Boca	-2	-1	-1	-1	-1	2	-1	0	0	-1	-1	-2	3	-0,08	10
C. Cordoba	1	3	0	1	2	-2	-1	1	1	0	1	-2	3	0,4	14
Colon	-2	0	0	-1	1	-2	0	-1	-1	1	0	-2	2	-0,12	8
DyJ	-1	-1	0	0	0	2	3	2	1	0	-1	-3	3	0,24	15
Estudiantes	0	1	2	1	1	-2	3	0	0	1	1	-2	3	0,56	18
Gimnasia	-1	-3	-1	-1	1	0	0	0	0	-1	-2	-3	2	-0,24	8
Godoy Cruz	0	-3	1	0	0	-1	2	0	0	2	0	-3	2	-0,24	9
Huracan	-1	1	0	0	1	-1	-3	-1	2	0	0	-3	3	0,08	12
Independiente	1	-2	-1	-2	0	-1	1	0	0	0	-1	-2	2	-0,36	6
Lanus	2	0	1	1	0	2	2	3	0	-1	1	-2	3	0,68	21
Newells	0	3	1	0	0	0	-2	-1	0	0	-1	-2	3	-0,4	8
Patronato	1	-1	0	1	-1	-1	-2	1	0	0	2	-2	2	0,04	10
Platense	-1	1	0	0	2	2	1	-3	-2	-2	-1	-3	3	0,36	19
Racing	-1	-1	1	0	0	1	0	0	0	0	0	-3	1	-0,16	6
River	1	2	0	0	-1	1	-1	0	0	0	-2	-2	2	-0,48	6
R. Central	0	1	0	0	2	-2	1	1	0	1	0	-2	2	-0,12	12
San Lorenzo	2	-2	-1	0	0	-1	0	-1	1	0	1	-2	2	-0,24	9
Sarmiento	-2	-1	1	0	0	2	1	-2	-1	-3	2	-3	3	-0,04	15
Talleres	-2	0	0	-1	-1	1	1	0	0	3	1	-3	3	-0,12	10
Union	1	1	-1	2	-2	0	-3	3	-1	0	0	-3	3	-0,04	13
Velez	0	3	0	1	0	1	-1	0	0	0	1	-2	3	0,36	14

Cuadro 4.4: LPF 2021 - Diferencias en los descansos - Fechas 15 a 25

Además, como remarcamos antes, la máxima diferencia que se da en el torneo es de tres días. Si bien podríamos fijar las cotas $u_{i,j}$ a 3 para cada partido, lo que se hizo fue fijarlas a 2 cuando las fechas son de fin de semana a fin de semana y a 1 cuando son de fin de semana a día de semana o viceversa. Esto lo hacemos debido a que las fechas que van de fin de semana a fin de semana tienen mayor descanso y una diferencia de un día más no es tan significativa. De esta manera, la

Fecha	Fecha 1	Fecha 2	Fecha 3	Fecha 4	Fecha 5	Fecha 6	Fecha 7	Fecha 8	Fecha 9	Fecha 10	Fecha 11	Fecha 12	Fecha 13	Fecha 14
Aldosivi	0	8	3	4	5	9	6	3	6	6	10	3	7	6
Argentinos	0	6	3	5	8	5	8	4	3	6	9	6	7	5
Arsenal	0	6	3	5	8	4	9	3	5	5	11	3	6	9
Atl. Tucumán	0	6	4	5	4	9	6	3	6	5	10	5	7	6
Banfield	0	6	4	4	7	7	8	4	3	6	9	6	7	6
Boca	0	6	4	4	9	4	9	3	4	6	9	6	6	8
C. Córdoba	0	7	3	5	8	4	9	3	4	6	10	4	6	10
Colon	0	6	3	4	7	7	7	3	6	6	10	3	7	7
DyJ	0	7	4	4	4	10	5	3	6	5	9	6	8	5
Estudiantes	0	9	3	4	8	4	9	4	3	6	9	6	6	6
Gimnasia	0	8	4	4	4	8	7	4	5	6	9	6	5	6
Godoy Cruz	0	6	3	3	7	7	7	4	3	8	10	3	7	7
Huracan	0	6	3	4	7	7	7	4	5	6	10	3	6	9
Independiente	0	7	3	4	7	7	6	3	6	6	10	3	7	7
Lanus	0	6	3	4	7	7	6	5	5	7	9	3	6	10
Newells	0	6	4	4	7	5	8	3	6	6	10	3	6	8
Patronato	0	8	3	4	8	4	9	4	3	5	10	6	6	6
Platense	0	8	3	4	7	5	9	3	4	5	10	6	6	6
Racing	0	8	3	4	7	5	9	3	4	6	9	6	7	5
River	0	6	3	4	7	7	7	3	6	6	10	3	6	9
R. Central	0	7	3	3	7	7	6	4	4	6	9	6	7	5
San Lorenzo	0	6	3	4	7	7	8	4	3	5	10	6	8	5
Sarmiento	0	9	3	4	7	8	5	3	6	7	9	3	7	6
Talleres	0	6	3	4	7	7	8	4	3	5	10	6	7	6
Union	0	7	3	4	7	7	8	3	4	6	9	7	5	6
Velez	0	9	3	4	5	7	6	5	3	7	10	4	7	7

Cuadro 4.5: Modelo 2021 - Días de descanso - Fechas 1 a 14

solución final no tendrá diferencia de 3 días en los equipos. La máxima cantidad de partidos por día se considera como seis para todos los días y cuatro a la cantidad de equipos populares que pueden jugar en un mismo día. Fijamos ciertos partidos importantes a jugarse en los domingos debido a requerimientos de la organización. Por otro lado, se tuvieron en cuenta los partidos jugados en copas internacionales (Copa Sudamericana y Copa Libertadores) y Copa Argentina para que los equipos participantes tengan tres días de descanso antes o después de cada juego.

En las tablas 4.5 y 4.6 podemos ver los días de descanso que tiene cada equipo a lo largo del torneo y, al igual que antes, podemos ver en la segunda tabla el valor mínimo, máximo y el promedio de descansos por equipo durante el torneo. Observemos que todos los equipos en alguna fecha descansan tan solo 3 días hasta su siguiente partido. Este valor es similar al visto en la respectiva tabla de descansos del fixture dado por la LPF. Pero, por otro lado, podemos ver un leve aumento en el máximo de días que descansa cada equipo (en la mayoría de los equipos). Este último punto, si bien no es fundamental para el objetivo que estamos buscando, sí es una característica positiva del fixture, dado que cuanto más descanso tengan los equipos, mejor es. En cuanto a la última columna, podemos ver que casi no existe diferencia en los promedios de descansos de cada uno de los equipos. Cabe destacar que en ambas tablas la columna con el máximo valor de descanso viene dada por la fecha 21, debido a que hubo 10 días entre que terminó una fecha y comenzó la siguiente.

A continuación, derivadas de las tablas anteriores, obtenemos las diferencias en los descansos para cada equipo a lo largo del torneo. Esto se presenta en las tablas 4.7 y 4.8.

Fecha	Fecha 15	Fecha 16	Fecha 17	Fecha 18	Fecha 19	Fecha 20	Fecha 21	Fecha 22	Fecha 23	Fecha 24	Fecha 25	Mínimo	Máximo	Promedio
Aldosivi	8	7	3	4	7	7	12	5	4	6	8	3	12	5,88
Argentinos	8	8	3	4	7	8	11	5	3	6	10	3	11	5,92
Arsenal	8	5	3	4	7	6	14	4	5	7	5	3	14	5,8
Atl. Tucuman	9	6	4	4	4	7	15	3	5	8	4	3	15	5,8
Banfield	9	5	4	4	7	6	12	5	4	8	5	3	12	5,84
Boca	8	4	4	5	6	7	12	5	4	8	6	3	12	5,88
C. Cordoba	7	4	4	5	6	6	14	4	5	7	6	3	14	5,88
Colon	9	6	3	5	4	7	16	3	5	7	5	3	16	5,84
DyJ	7	8	4	4	4	7	14	4	6	7	6	3	14	5,88
Estudiantes	8	8	4	4	4	9	13	4	3	7	7	3	13	5,92
Gimnasia	7	8	3	5	6	7	12	6	3	7	7	3	12	5,88
Godoy Cruz	9	4	4	5	7	7	12	5	4	8	4	3	12	5,76
Huracan	8	4	5	4	8	7	12	5	5	6	4	3	12	5,8
Independiente	6	8	3	6	4	7	14	4	4	8	6	3	14	5,84
Lanus	7	5	4	4	7	7	13	5	4	7	6	3	13	5,88
Newells	7	7	3	6	4	7	14	4	4	8	5	3	14	5,8
Patronato	8	8	4	4	4	8	14	5	3	6	7	3	14	5,88
Platense	8	7	5	4	4	7	15	5	5	6	4	3	15	5,84
Racing	8	7	4	5	4	7	16	3	5	7	4	3	16	5,84
River	8	6	3	4	5	8	14	4	5	7	4	3	14	5,8
R. Central	8	8	4	4	4	7	15	4	5	7	4	3	15	5,76
San Lorenzo	9	6	3	4	7	6	12	5	4	8	5	3	12	5,8
Sarmiento	8	7	3	6	5	6	13	5	4	7	9	3	13	6
Talleres	9	6	3	4	8	6	12	5	3	7	7	3	12	5,84
Union	8	7	4	5	4	7	14	4	5	7	6	3	14	5,88
Velez	7	7	3	5	7	6	13	5	3	7	6	3	13	5,84

Cuadro 4.6: Modelo 2021 - Días de descanso - Fechas 15 a 25

El modelo devuelve un fixture óptimo cuya función objetivo es de 124. Es decir, el total de diferencias de descanso a lo largo del torneo se mejoró en 176 días, lo cual equivale a una mejora de aproximadamente el 58 %.

Fecha	Fecha 1	Fecha 2	Fecha 3	Fecha 4	Fecha 5	Fecha 6	Fecha 7	Fecha 8	Fecha 9	Fecha 10	Fecha 11	Fecha 12	Fecha 13	Fecha 14
Aldosivi	0	0	0	-1	-1	0	1	0	-1	0	0	1	-2	0
Argentinos	0	0	0	0	1	2	0	0	1	0	1	0	0	0
Arsenal	0	0	0	-1	0	1	0	1	1	0	-1	0	0	0
Atl. Tucumán	0	0	1	-1	0	0	-1	0	0	0	0	1	0	-1
Banfield	0	0	-1	0	0	-2	1	0	1	0	0	0	0	1
Boca	0	0	-1	0	-1	0	0	0	0	0	0	-1	1	1
C. Córdoba	0	-1	0	0	0	0	0	0	0	-1	0	-1	0	0
Colon	0	0	0	-1	-2	1	-1	0	0	0	1	1	-1	-1
DyJ	0	-1	-1	0	1	-2	1	0	-1	1	0	0	0	1
Estudiantes	0	-2	1	1	0	0	-1	0	0	0	0	0	0	-1
Gimnasia	0	0	-1	0	0	-1	0	1	0	2	1	1	2	0
Godoy Cruz	0	1	1	1	0	0	0	-1	0	-2	0	0	0	1
Huracan	0	0	0	0	0	0	0	1	0	0	-1	0	0	0
Independiente	0	2	0	0	0	0	-1	0	0	0	-1	0	0	0
Lanus	0	0	0	0	0	0	0	-1	1	0	1	0	0	0
Newells	0	0	-1	0	0	0	-2	0	0	0	0	0	0	-1
Patronato	0	1	0	0	0	0	0	0	0	0	-1	0	-1	0
Platense	0	0	0	0	0	-1	0	0	-1	0	0	0	0	0
Racing	0	0	0	0	0	0	0	0	0	0	1	0	0	1
River	0	1	0	0	0	0	0	0	0	0	0	0	0	-1
R. Central	0	2	1	1	0	0	2	-1	0	0	0	0	0	0
San Lorenzo	0	1	1	0	0	0	0	0	0	0	-1	0	0	1
Sarmiento	0	-1	0	0	0	2	1	0	0	0	1	0	0	0
Talleres	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
Union	0	-1	1	0	0	0	0	1	-1	1	0	-1	1	0
Velez	0	-2	1	1	2	0	0	-1	0	-1	-1	-1	0	0

Cuadro 4.7: Modelo 2021 - Diferencias en los descansos - Fechas 1 a 14

En cuanto al análisis por equipo, como es de esperar, al disminuir tanto la can-

Fecha	Fecha 15	Fecha 16	Fecha 17	Fecha 18	Fecha 19	Fecha 20	Fecha 21	Fecha 22	Fecha 23	Fecha 24	Fecha 25	Mínimo	Máximo	Promedio	Descansos totales
Aldosivi	0	1	0	0	-1	0	0	0	-1	0	-1	-2	1	-0,2	3
Argentinos	-1	-2	0	0	0	-1	1	0	0	0	-1	-2	2	0,04	6
Arsenal	1	2	0	0	-2	1	0	0	0	1	0	-2	2	0,16	8
Atl. Tucuman	0	2	0	0	0	0	0	1	0	-1	0	-1	2	-0,04	4
Banfield	-1	0	1	1	0	1	0	0	0	0	0	-2	1	0,08	6
Boca	-1	0	0	0	0	0	1	-1	0	-1	0	-1	1	-0,12	3
C. Cordoba	1	0	-1	0	1	0	0	0	0	0	0	-1	1	-0,08	2
Colon	0	0	0	-1	0	0	0	1	1	1	1	-2	1	0	7
DyJ	1	0	0	0	0	0	0	-1	-1	0	0	-2	1	-0,08	5
Estudiantes	0	0	0	1	1	-2	-1	1	0	0	1	-2	1	-0,04	6
Gimnasia	-1	-1	0	0	0	-1	0	-1	0	0	0	-1	2	0,04	7
Godoy Cruz	-1	0	0	-1	0	-1	-1	-1	-1	-2	0	-2	1	-0,28	4
Huracan	-1	0	-1	0	0	1	1	0	0	1	0	-1	1	0,04	4
Independiente	1	-1	0	-1	0	-1	0	1	0	0	1	-1	2	0	5
Lanus	1	0	-1	0	0	2	1	0	1	0	0	-1	2	0,2	7
Newells	0	1	0	0	0	0	0	0	0	0	0	-2	1	-0,12	1
Patronato	0	0	0	0	0	0	-1	0	1	1	0	-1	1	0	3
Platense	0	0	-1	0	0	0	-1	0	0	2	2	-1	2	0	4
Racing	0	0	0	-1	0	0	0	1	-1	-1	0	-1	1	0	3
River	1	0	0	0	-1	0	1	-1	0	0	0	-1	1	0	3
R. Central	0	0	1	1	0	0	0	-1	0	0	0	-1	2	0,24	8
San Lorenzo	0	0	1	0	0	0	0	0	0	0	0	-1	1	0,12	4
Sarmiento	0	0	0	0	2	0	-1	0	0	0	1	-1	2	0,2	7
Talleres	0	0	0	0	0	1	1	1	1	0	-1	-1	1	0,08	4
Union	0	0	0	1	0	0	0	0	0	0	-1	-1	1	0,04	5
Velez	0	-2	1	0	0	0	-1	0	0	-1	-2	-2	2	-0,28	5

Cuadro 4.8: Modelo 2021 - Diferencias en los descansos - Fechas 15 a 25

tidad total de diferencias en los descansos, disminuye la cantidad de descansos que tiene cada uno de los equipos por separado en todo el torneo. Además, logramos limitar la máxima diferencia en los descansos a 2, a diferencia del torneo creado por la LPF que tiene diferencias de 3 días.

Por ejemplo, podemos destacar el caso de Lanus. Dicho equipo, a lo largo del torneo, tiene una ventaja de 21 días con respecto a distintos rivales, lo cual le otorga una notable ventaja a la hora de llegar a los partidos con menos cansancio. Este valor fue reducido a 7 por el modelo, generando una mejora de más del 65 %. El resto de las comparaciones de los descansos de los equipos entre el torneo original y el arrojado por el modelo pueden verse en el anexo 6.2.

Pero como no todas las fechas son iguales, decidí setear distintas cotas a las diferencias en los descansos, de acuerdo a la distancia entre la presente fecha y la anterior. Esto es, aquellas fechas que transcurren durante el fin de semana y cuya fecha anterior también ocurrió durante el fin de semana, fijé el valor de la cota a 2 días de diferencia. Por otro lado, aquellas fechas que se jugaron durante un fin de semana y la fecha anterior se realizó durante la semana recién transcurrida, la cota fue bajada a 1, puesto que al haber tan pocos días de diferencia entre las fechas, un día de diferencia en los descansos es mucho más perjudicial que un día más de descanso al jugar entre fines de semana. Lo mismo sucede cuando una fecha se juega durante la semana y la anterior se jugó durante el fin de semana anterior. Vale aclarar que no hay fechas entre semana cuya fecha anterior también fue entre semana.

Por último, podemos observar que los promedios de diferencias de descansos por equipo son más parejos con el fixture arrojado con el modelo. Hay una gran cantidad de equipo cuyo fixture está equilibrado en cuanto a los diferencias a favor y en contra que tienen durante el torneo. Para todos aquellos equipos cuyo promedio no es 0, vemos una gran baja en sus valores con respecto a los dados en el fixture utilizado.

Además, el intervalo en el que se mueven los promedios es mucho menor que el original ([-0.48, 0,68] contra [-0.28, 0.24])

La mejora en las diferencias en los descansos entre ambos fixtures es notoria. Sin embargo, pueden existir restricciones de seguridad, televisivas o de organización que puedo no estar teniendo en cuenta, lo cual tendería a aumentar el valor de la función objetivo. De todas formas, la diferencia es tan amplia que el resultado probablemente continúe siendo mejor.

Las cotas seleccionadas para correr el modelo resultaron de una serie de pruebas que concluyeron en los valores dichos arriba. El modelo resulta infactible al fijar la cota superior de las diferencias en los descansos a 1 para todas las fechas. Por eso, preferí fijar dicho valor para las fechas con menos descansos y elevar a 2 el resto.

4.2. Temporada 2022

La siguiente sección corresponde al torneo de la LPF de este año. En este caso, el torneo consta de 28 equipos con el mismo formato que el anterior. Comenzamos analizando el armado del fixture que se utilizó en el torneo recientemente terminado. Debido a la amplia cantidad de equipos que hay en la primera división del fútbol argentino y el recortado tiempo de competencia debido a la copa mundial que se juega en Noviembre (momento en el cual todo torneo de fútbol debe suspenderse), el calendario del año 2022 se vuelve más complicado de asignar dado que existen pocas semanas para tantos torneos. Recordemos que los equipos participan en copas internacionales y la Copa Argentina (además de la competencia mencionada). Por ejemplo, la fecha 19 del torneo se jugó a lo largo de 4 días. El equipo Velez participaba en la Copa Libertadores y en la Copa Argentina durante dicha fecha, y le tocaba jugar en ambas. Debido a la restricción de 3 días de descanso entre cada partido, el partido de la copa internacional y el partido de la Copa Argentina, Velez no puede jugar la fecha 19 sin romper dicha restricción. Por esta razón, la organización decidió posponer su partido hasta el final del torneo.

Dado que el torneo consta de 27 fechas, todos los equipos tienen entre 13 y 14 partidos de visitante y de local. En cuanto a los *breaks*, nuevamente dos equipos concluyen el torneo con 0 *breaks* en su calendario. Además, la gran mayoría de los equipos tienen 1 *break* de local y uno de visitante, haciendo el torneo mucho más parejo. Tan solo dos equipos tienen 2 *breaks* de local y un único equipo tiene 2 *breaks* de visitante. Por último, relacionado a este aspecto, a diferencia del torneo de 2021, ningún equipo tiene un *break* en las primeras ni las últimas dos fechas del torneo.

Con el fin de no llenar el capítulo de tablas, se puede encontrar en el capítulo 6 la tabla con los descansos de los equipos para el torneo de la LPF 2022 (6.1).

A continuación, se presentan las tablas 4.9 y 4.10 con las diferencias en los descansos para el torneo analizado. Análogamente a lo presentado para el torneo anterior, la segunda tabla presenta el valor mínimo, máximo y el promedio de las diferencias para cada equipo. El valor total de las diferencias a lo largo del torneo (función objetivo del modelo) resultó ser 336 días. A primera vista, podemos observar

que en este torneo existe un partido en el cual la diferencia en los descansos de los participantes fue 4 días (coloreado con naranja). Al igual que lo sucedido en 2021, en varios partidos la diferencia fue de 3 días.

Fecha	Fecha 1	Fecha 2	Fecha 3	Fecha 4	Fecha 5	Fecha 6	Fecha 7	Fecha 8	Fecha 9	Fecha 10	Fecha 11	Fecha 12	Fecha 13	Fecha 14	Fecha 15
Aldosivi	0	-1	0	2	-1	-2	-2	-1	1	2	0	-1	-2	3	0
Argentinos	0	-1	-1	-2	-2	-1	1	-2	0	2	0	-1	0	0	0
Arsenal	0	-1	-2	0	2	-1	0	0	1	2	-1	-2	1	2	0
Atl. Tucumán	0	1	-1	-1	1	-2	0	1	0	1	1	2	2	2	1
Banfield	0	1	1	0	0	0	2	0	-1	-2	-1	1	1	2	-1
Barracas C	0	2	1	0	0	2	-1	2	1	0	-2	0	2	0	-1
Boca	0	-2	-1	0	0	0	2	0	0	1	2	0	0	-1	1
C. Córdoba	0	2	-1	-1	-3	1	3	-1	1	0	3	0	-2	0	1
Colon	0	1	-1	0	0	2	0	-1	-1	0	1	-2	-1	0	1
DyJ	0	1	0	2	0	-1	2	-3	1	1	1	0	0	0	-1
Estudiantes	0	1	1	2	0	1	-3	1	-1	-1	1	0	-1	0	0
Gimnasia	0	-1	2	0	0	1	0	1	1	-1	1	-1	1	0	0
Godoy Cruz	0	0	-2	-2	-1	-2	1	0	0	0	0	1	2	1	0
Huracan	0	2	0	1	0	0	0	-1	0	0	-1	1	-2	1	1
Independiente	0	1	1	-2	-2	-1	-1	-2	-1	-1	-1	-1	-2	-1	0
Lanus	0	-1	1	0	0	-1	0	0	0	1	0	0	2	0	0
Newells	0	0	1	2	0	2	0	1	0	-1	-1	2	0	0	-1
Patronato	0	1	0	0	2	-2	0	1	3	0	-2	-1	2	2	0
Platense	0	0	-2	-2	-3	1	0	0	-1	1	2	0	-1	0	2
Racing	0	0	0	0	1	0	1	-1	-1	0	1	0	0	-2	-2
River	0	-1	1	0	0	0	-1	1	-1	-2	-1	1	0	-2	-1
R. Central	0	-2	2	0	0	2	-1	2	0	-2	-3	-1	-2	1	1
San Lorenzo	0	0	2	1	-2	-2	-2	0	0	1	0	0	-2	0	2
Sarmiento	0	1	-1	0	3	0	1	3	0	0	1	0	2	-1	2
Talleres	0	-1	-1	0	3	1	1	0	1	-1	0	1	1	-2	-2
Tigre	0	-2	1	0	2	-1	-1	-1	-3	-1	-1	1	0	-2	-1
Union	0	-1	-1	0	0	1	-2	1	0	0	0	0	0	0	0
Velez	0	0	0	0	0	2	0	-1	0	0	0	0	-1	-3	-2

Cuadro 4.9: LPF 2022- Diferencias en los descansos - Fechas 1 a 14

Fecha	Fecha 16	Fecha 17	Fecha 18	Fecha 19	Fecha 20	Fecha 21	Fecha 22	Fecha 23	Fecha 24	Fecha 25	Fecha 26	Fecha 27	Mínimo	Máximo	Promedio	Total
Aldosivi	-1	0	0	3	1	1	-1	-1	2	0	1	1	-2	3	0,1481	17
Argentinos	-1	0	0	3	0	-1	-2	0	0	-1	-1	0	-2	3	-0,3704	6
Arsenal	-1	1	-2	0	-1	-2	0	-1	-1	2	-2	1	-2	2	-0,1852	12
Atl. Tucuman	-2	-1	0	1	0	0	-1	-1	0	0	1	1	-2	2	0,2222	15
Banfield	-1	1	-1	1	0	1	0	1	2	0	0	1	-2	2	0,2963	15
Barracas C	-3	-1	-1	0	0	1	0	1	0	1	2	-1	-3	2	0,1852	15
Boca	2	0	0	-2	-1	-2	2	-3	-2	-1	1	0	-3	2	-0,1481	11
C. Cordoba	0	0	4	1	0	-1	-2	0	-1	-2	-1	-1	-3	4	0	16
Colon	3	0	1	0	1	1	-1	2	0	2	0	-2	-2	3	0,2222	15
DyJ	1	0	-2	-3	0	1	1	-1	-2	-2	0	-1	-3	2	-0,1852	11
Estudiantes	1	-1	-1	-1	0	0	1	2	1	0	-1	-1	-3	2	0,0370	12
Gimnasia	-1	0	0	0	0	-2	0	3	-2	-1	1	-1	-2	3	0,0370	11
Godoy Cruz	-1	-1	-4	-1	0	2	0	0	2	1	0	1	-4	2	-0,1111	11
Huracan	1	0	2	0	1	-1	1	1	0	0	-2	-2	-2	2	0,1111	12
Independiente	1	0	0	0	1	-2	0	0	-1	-1	0	0	-2	1	-0,5556	4
Lanus	0	1	1	2	0	0	0	0	-1	-1	1	0	-1	2	0,1852	9
Newells	1	1	0	-3	-2	2	1	1	1	-1	-1	1	-3	2	0,2222	16
Patronato	-1	0	0	0	2	-1	1	-2	0	1	0	2	-2	3	0,2963	17
Platense	1	1	0	-1	0	2	-1	1	0	1	2	0	-3	2	0,1111	14
Racing	0	0	1	0	0	1	0	1	0	-2	-1	-2	-2	1	-0,1852	6
River	-1	1	0	-1	0	0	2	-2	0	-1	1	2	-2	2	-0,1852	9
R. Central	0	0	0	1	-2	-2	0	0	2	0	-1	2	-3	2	-0,1111	13
San Lorenzo	0	0	0	0	0	0	-1	0	0	1	0	-1	-2	2	-0,1111	7
Sarmiento	1	0	1	0	2	2	2	-1	0	1	0	-1	-1	3	0,6667	22
Talleres	0	0	2	-1	-1	0	0	-1	-2	-1	-1	1	-2	3	-0,1111	11
Tigre	1	-1	-2	1	2	2	1	1	1	1	1	-1	-3	2	-0,0741	15
Union	1	0	-1	1	-1	-1	-1	0	1	2	-1	1	-2	2	-0,0370	8
Velez	-1	-1	2	-1	-2	-1	-2	-1	0	1	1	0	-3	2	-0,3703	6

Cuadro 4.10: LPF 2022- Diferencias en los descansos - Fechas 15 a 27

Para correr el modelo en el caso de 2022, se mantuvieron los mismos parámetros que el torneo pasado. Se separaron las fechas en dos conjuntos disjuntos: por un lado

las fechas que se jugaron durante el fin de semana y la anterior fue fin de semana, y todas las otras fechas. Para el primer conjunto de fechas la cota superior en los descansos continua siendo 2, mientras que en el resto de las fechas la diferencia en los descansos puede ser de tan solo un día (por la misma razón antes explicada).

Los equipos populares se mantienen como los 5 denominados grandes, con la restricción de que solo 4 de ellos pueden jugar el mismo día. En cada uno de los días de cada fecha se deben jugar al menos 2 partidos y como máximo 6.

Nuevamente, el modelo genera una amplia mejora en los descansos de los equipos, y por ende en las diferencias de los descansos, durante el torneo. Podemos ver en las tablas 4.11 y 4.12 podemos ver los descansos de los equipos a lo largo del torneo.

De acuerdo a los descansos de la LPF 2022 (6.1 y 6.2) el mínimo de cada equipo es prácticamente el mismo (Velez tenía un descanso de 2 días pero dicho partido fue reprogramado y Barracas Central nunca descansa 3 días). En cuanto a los máximos, todos los equipos varían entre 8, 9 y 10 días de descanso. En este aspecto, el modelo arroja un fixture con características similares, donde todos los equipos tienen un mínimo de 3 días de descanso y los máximos varían en los mismos valores. Por último, podemos observar que los promedios del modelo son levemente superiores en todos los equipos a los del fixture original.

Como dato curioso, notemos por los colores que la mayoría de los *breaks* de este torneo se fijaron sobre las fechas 25 y 26.

Fecha	Fecha 1	Fecha 2	Fecha 3	Fecha 4	Fecha 5	Fecha 6	Fecha 7	Fecha 8	Fecha 9	Fecha 10	Fecha 11	Fecha 12	Fecha 13	Fecha 14	Fecha 15
Aldosivi	0	5	4	6	7	7	7	4	5	6	3	8	7	5	4
Argentinos	0	4	5	4	7	7	7	7	3	5	6	8	6	4	6
Arsenal	0	7	5	3	7	9	6	5	4	6	6	7	5	4	5
Atl. Tucumán	0	7	5	4	6	7	8	4	5	5	6	7	5	4	5
Banfield	0	7	5	4	6	7	7	7	4	4	8	5	8	3	3
Barracas C	0	9	4	3	6	6	8	7	3	5	8	5	7	3	5
Boca	0	8	4	4	6	7	6	8	3	5	8	5	8	3	4
C. Córdoba	0	8	4	3	7	7	8	5	4	5	7	7	5	4	5
Colon	0	8	4	3	7	7	8	5	4	6	4	8	5	4	5
DyJ	0	5	5	4	6	7	8	3	6	5	6	7	5	4	5
Estudiantes	0	5	4	4	6	10	6	6	3	5	8	7	6	3	5
Gimnasia	0	5	4	7	5	7	8	5	4	6	3	7	8	5	4
Godoy Cruz	0	6	4	5	6	7	8	6	4	5	3	7	8	5	4
Huracan	0	6	5	4	6	7	7	6	5	5	3	8	6	6	4
Independiente	0	7	4	4	8	8	6	4	6	5	5	8	4	7	4
Lanus	0	6	5	3	7	7	7	7	3	6	3	8	5	6	4
Newells	0	7	5	3	6	10	6	5	4	6	6	7	5	5	5
Patronato	0	5	4	4	8	8	6	5	4	5	8	5	8	3	6
Platense	0	6	3	6	7	7	6	5	4	4	9	5	8	3	5
Racing	0	6	3	7	6	7	6	5	4	5	8	5	8	3	4
River	0	5	5	3	7	7	8	6	3	6	4	8	5	4	5
R. Central	0	6	4	6	5	8	7	3	5	6	6	7	6	4	3
San Lorenzo	0	7	5	3	7	6	7	8	5	4	5	9	6	3	4
Sarmiento	0	5	5	4	9	7	7	3	6	6	4	7	6	6	4
Talleres	0	6	5	3	7	7	7	7	4	5	4	9	7	3	5
Tigre	0	9	3	4	6	7	7	7	3	4	9	6	5	4	5
Union	0	9	4	3	6	8	6	7	5	5	3	8	7	5	4
Velez	0	6	3	7	5	7	8	6	3	6	3	8	7	5	4

Cuadro 4.11: Modelo 2022 - Días de descanso - Fechas 1 a 14

Como en los casos anteriores, de la tabla de descansos se derivan las tablas con las diferencias en los descansos para cada equipo. Estas dos pueden verse en 4.13 y 4.14.

Fecha	Fecha 16	Fecha 17	Fecha 18	Fecha 19	Fecha 20	Fecha 21	Fecha 22	Fecha 23	Fecha 24	Fecha 25	Fecha 26	Fecha 27	Mínimo	Máximo	Promedio
Aldosivi	6	4	8	4	5	7	4	4	5	6	3	5	3	8	5.3462
Argentinos	3	9	5	4	4	6	9	3	3	4	5	5	3	9	5.3462
Arsenal	6	7	7	4	4	7	6	3	6	3	5	4	3	9	5.4231
Atl. Tucuman	5	9	7	3	3	6	9	3	5	3	5	6	3	9	5.4615
Banfield	6	9	7	3	5	4	9	3	3	6	3	6	3	9	5.4615
Barracas C	6	8	6	5	4	5	6	3	6	5	4	4	3	9	5.4231
Boca	5	10	6	4	4	4	8	3	4	4	6	5	3	10	5.4615
C. Cordoba	6	7	7	4	4	7	5	4	5	4	5	4	3	8	5.4231
Colon	6	9	6	3	3	6	9	4	3	4	5	7	3	9	5.5
DyJ	5	9	6	3	4	8	6	3	6	3	5	6	3	9	5.3846
Estudiantes	4	9	4	5	4	6	9	4	3	4	4	6	3	10	5.3846
Gimnasia	6	6	7	4	4	6	6	4	4	4	5	6	3	8	5.3846
Godoy Cruz	5	7	7	5	4	4	9	3	3	4	4	7	3	9	5.3846
Huracan	5	7	7	5	4	4	9	3	3	5	3	7	3	9	5.3846
Independiente	5	7	6	3	6	7	6	3	5	5	3	6	3	8	5.4615
Lanus	6	7	7	5	4	4	9	3	4	3	4	6	3	9	5.3462
Newells	5	7	7	3	5	7	6	3	5	3	6	4	3	10	5.4231
Patronato	3	8	6	4	4	8	7	4	3	3	4	7	3	8	5.3846
Platense	4	9	5	4	5	8	5	4	4	4	3	6	3	9	5.3462
Racing	5	9	4	5	5	8	5	3	6	3	3	7	3	9	5.3846
River	6	8	7	3	4	5	9	4	3	4	3	7	3	9	5.3462
R. Central	6	9	5	4	4	9	5	3	6	3	3	9	3	9	5.4615
San Lorenzo	5	9	7	3	4	5	9	3	3	4	5	5	3	9	5.4231
Sarmiento	6	4	8	3	6	7	5	4	5	3	5	6	3	9	5.4231
Talleres	4	9	6	4	3	6	9	3	3	6	3	6	3	9	5.4231
Tigre	6	7	7	5	5	4	5	5	5	3	6	4	3	9	5.4231
Union	6	6	6	4	5	8	5	3	6	3	5	4	3	9	5.4231
Velez	5	7	7	4	6	4	7	4	3	6	3	5	3	8	5.3462

Cuadro 4.12: Modelo 2022 - Días de descanso - Fechas 15 a 27

El modelo obtiene un fixture con un total de 146 diferencias en los descansos, en comparación a los 336 días que tuvo el fixture original. Es decir, generó una reducción del 56 %. Además, notemos que todos los partidos tienen a lo sumo 2 días de diferencia, cuando antes había 3 y hasta un partido con 4 días. Más aún, como restringimos en el modelo, las fechas que tienen 2 días de distancia pertenecen a fechas en fin de semana cuya fecha anterior también se jugó durante fin de semana.

Por otro lado, se puede ver que la cantidad total de discrepancias por equipo a lo largo del torneo disminuyó notablemente, haciendo que el torneo tenga más paridad. El equipo Sarmiento, por ejemplo, tuvo 22 días de diferencia a favor en torneo jugado, pero tendría tan solo 4 en el torneo arrojado por el modelo. Dicho equipo ocupa el puesto con la mayor cantidad de días de ventaja, siendo el mínimo de 4 días (Independiente). Por lo contrario, el mínimo de ventaja que tiene un equipo en el fixture del modelo es 1 día, mientras que el máximo llega tan solo a 11. Todos los valores totales de los equipos están comparados en el anexo 6.3.

Por último, analicemos la columna de promedios. Cuanto más cerca está el promedio del 0, mejor es. Los promedios del fixture dado por la LPF varían entre -0.5556 y 0.6667, lo cual se refleja en la variabilidad que existe en la cantidad total de descansos. En el fixture dado por el modelo, los promedios están acotados inferiormente por -0.2308 y superiormente por 0.2308. Observar que el rango de variabilidad es mucho más acotado, generando más paridad.

Todas estas reducciones de días en los descansos son fundamentales para los equipos. Quizás no generen un ahorro desde el aspecto económico (o quizás sí), pero es muy importante para el cuidado de los jugadores. Esto último repercute directamente en el rendimiento del equipo, ya sea por el desgaste de los jugadores o el hecho de perder jugadores por lesión.

Fecha	Fecha 1	Fecha 2	Fecha 3	Fecha 4	Fecha 5	Fecha 6	Fecha 7	Fecha 8	Fecha 9	Fecha 10	Fecha 11	Fecha 12	Fecha 13	Fecha 14	Fecha 15
Aldosivi	0	0	0	0	-1	1	1	0	-1	0	0	0	1	0	0
Argentinos	0	1	-1	-1	0	0	0	0	0	-1	-1	1	1	-1	0
Arsenal	0	0	0	0	0	1	0	0	0	0	0	0	0	0	-1
Atl. Tucumán	0	-2	0	0	0	0	0	0	1	0	0	0	0	0	0
Banfield	0	0	-1	0	0	0	-1	1	0	1	0	0	0	0	0
Barracas C	0	0	0	1	0	0	-1	0	0	0	1	0	-1	1	0
Boca	0	0	-1	-1	0	0	1	-1	0	0	0	0	0	1	1
C. Córdoba	0	0	1	0	0	0	-2	0	0	0	-1	0	0	0	0
Colon	0	1	1	0	-1	0	0	0	1	0	1	-1	0	0	0
DyJ	0	1	0	1	-1	0	-1	0	0	1	0	0	0	-1	-1
Estudiantes	0	0	1	0	0	-1	2	1	0	0	0	2	1	1	-1
Gimnasia	0	0	-1	0	0	0	0	0	-1	0	0	0	-1	0	0
Godoy Cruz	0	0	0	-1	0	0	0	1	1	0	0	0	-1	1	0
Huracan	0	0	0	0	1	0	0	1	-1	1	0	0	0	1	1
Independiente	0	-1	1	0	0	-1	0	-1	0	0	-1	0	1	-1	0
Lanus	0	-1	0	0	0	1	0	-1	0	0	0	-1	-1	-1	1
Newells	0	0	0	1	0	-2	0	0	1	-1	0	1	0	1	-1
Patronato	0	0	0	0	0	2	0	0	-1	0	0	0	-2	0	0
Platense	0	0	1	0	2	1	0	0	0	0	-1	0	0	0	0
Racing	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
River	0	2	-1	0	0	0	0	0	1	0	0	0	0	0	0
R. Central	0	0	0	1	0	-1	0	1	-1	0	1	-1	1	-1	0
San Lorenzo	0	0	0	0	-1	0	-1	-1	0	1	1	-2	2	0	0
Sarmiento	0	-1	-1	0	-2	0	0	0	-1	0	0	1	0	-1	0
Talleres	0	1	0	0	0	0	1	1	0	-1	-1	-1	-1	0	0
Tigre	0	0	1	0	1	0	0	-1	1	0	-1	1	0	0	0
Union	0	-1	0	0	0	-1	1	-1	0	0	1	0	-1	0	1
Velez	0	0	0	-1	1	0	0	0	0	-1	0	0	1	0	0

Cuadro 4.13: Modelo 2022 - Diferencias en los descansos - Fechas 1 a 14

Fecha	Fecha 16	Fecha 17	Fecha 18	Fecha 19	Fecha 20	Fecha 21	Fecha 22	Fecha 23	Fecha 24	Fecha 25	Fecha 26	Fecha 27	Mínimo	Máximo	Promedio	Total
Aldosivi	0	0	-2	-1	-1	0	1	-1	-1	0	0	0	-2	1	-0,1538	4
Argentinos	1	0	0	-1	-1	0	0	0	0	0	0	0	-1	1	-0,1154	4
Arsenal	-1	0	0	0	1	0	0	0	0	1	-1	0	-1	1	0	3
Atl. Tucuman	0	0	0	1	1	0	-2	1	1	0	0	0	-2	1	0,0385	5
Banfield	-1	0	-1	0	-1	0	0	1	1	0	0	0	-1	1	-0,0385	4
Barracas C	0	0	1	0	0	-1	0	1	-1	0	1	0	-1	1	0,0769	6
Boca	0	-1	1	1	0	0	-1	1	1	-1	0	1	-1	1	0,0769	8
C. Cordoba	0	0	0	0	0	0	0	-1	0	-1	1	0	-2	1	-0,1154	2
Colon	0	1	1	0	0	0	0	0	1	-1	0	2	-1	2	0,2308	9
DyJ	1	0	0	1	0	0	-1	0	0	0	0	0	-1	1	0	5
Estudiantes	-1	0	0	-1	0	0	0	0	1	1	-1	1	-1	2	0,2308	11
Gimnasia	0	1	0	0	0	-2	0	-1	-1	0	0	0	-2	1	-0,2308	1
Godoy Cruz	0	0	0	0	0	0	0	0	0	0	0	-1	-1	1	0	3
Huracan	1	0	0	0	0	0	0	0	0	-1	0	0	-1	1	0,1538	6
Independiente	0	-1	2	0	-1	0	0	1	0	0	0	-1	-1	2	-0,1154	5
Lanus	0	0	-1	-1	1	1	0	0	-1	0	-1	0	-1	1	-0,1923	4
Newells	0	0	0	1	1	0	-1	0	0	0	0	0	-2	1	0,0385	6
Patronato	1	-2	-1	1	0	0	2	0	0	0	0	0	-2	2	0	6
Platense	-1	0	1	0	0	1	1	-1	-1	0	0	0	-1	2	0,1154	7
Racing	-1	0	0	-1	0	0	0	0	-1	1	1	0	-1	1	0,0385	4
River	0	0	-1	0	0	1	0	0	0	0	0	0	-1	2	0,0769	4
R. Central	-1	0	0	1	0	-1	0	0	0	0	0	-2	-2	1	-0,1154	5
San Lorenzo	1	0	0	0	0	-1	0	0	0	0	0	0	-2	2	-0,0385	5
Sarmiento	0	0	-2	0	-1	0	0	1	1	1	0	0	-2	1	-0,1923	4
Talleres	1	0	0	-1	0	-1	0	0	0	0	0	0	-1	1	-0,0769	4
Tigre	0	0	0	0	1	2	-1	-1	0	0	-1	0	-1	2	0,0769	7
Union	0	2	2	0	1	0	1	0	0	0	0	0	-1	2	0,1923	9
Velez	0	0	0	0	-1	1	1	-1	0	0	1	0	-1	1	0,0385	5

Cuadro 4.14: Modelo 2022 - Diferencias en los descansos - Fechas 15 a 27

4.3. Análisis de la corrida

Todas las corridas fueron hechas en la misma computadora con sistema operativo *Windows*, procesador INTEL Core i-5 y 12GB de RAM.

Tanto el modelo como los cálculos sobre los fixtures existentes se realizaron en *Python*, versión 3.10, y en el caso del modelo, optimiza llamando al solver *CPLEX*. Tanto el modelo, como los archivos input y los procesadores de los fixturesde la LPF pueden encontrarse en el repositorio https://github.com/NicoMarucho97/Tesis.git.

Ambos modelos corren en un tiempo razonable, por lo que, a pesar de fijar el tiempo máximo de corrida y dejar el límite de gap predeterminado, se conseguía el óptimo antes. El primer torneo tenía 26 equipos y, por lo tanto, 25 fechas. Luego de probar una serie de corridas, el resultado siempre era el mismo y los tiempos de corrida promediaban los 8 o 9 minutos. Por otro lado, el segundo torneo, que consta de 28 equipos y 26 fechas en un calendario más ajustado, obtenía el óptimo en unos 15 minutos aproximadamente en cada una de sus corridas.

Pero esto no fue siempre así. El modelo creado inicialmente tenía otras dificultades. Dicho modelo constaba de otros conjuntos de variables (binarias y enteras) que representaban las diferencias en los descansos. Por un problema en la formulación probablemente, el modelo inicial corría durante un largo tiempo sin poder llegar al óptimo. Durante la corrida, podía observarse en el output de las iteraciones que el modelo alcanzaba el óptimo entero, pero la relajación del problema obtenía soluciones muy lejanas a este último. El gap de dichas corridas no bajaba del 80 % a pesar de haber corrido durante dos días y de haber mantenido la misma solución entera desde los primeros minutos de corrida. Este problema fue resuelto implementando la sugerencia de [19].

Capítulo 5

Trabajo futuro

A lo largo del presente trabajo nos enfocamos en el mundo de la optimización matemática. Abordamos tanto los principales conceptos teóricos de la programación lineal como así el método más famoso para resolver dichos problemas, *SIMPLEX*. Además, dimos un pequeño apartado sobre programación lineal entera y el algoritmo de *Branch and Bound*, el cual usa el solver para resolver nuestro problema.

A continuación, ingresamos en un mundo bastante nuevo, pero con mucho por crecer en la matemática: Sports scheduling. El Sport Scheduling es el arte de armar calendarios, fixtures o cronogramas de distintos deportes a través de la programación de modelos matemáticos. En el capítulo correspondiente vimos distintos criterios para armar fixtures de fútbol que fueron estudiados con anterioridad.

Uno de dichos criterios es la diferencia en los descansos de los equipos, o como se lo conoce "*The Rest Difference Problem*". Este problema fue poco estudiado hasta la actualidad, pero es de suma importancia y repercute directamente en los equipos y los deportistas. Los torneos nacionales tienden a superponerse entre sí e incluso a hacerlo con los que son de escala internacional, produciendo que los jugadores deban jugar muchos partidos en poco tiempo. Por lo tanto, una diferencia en los descansos con respecto al equipo rival genera una desventaja notable en el rendimiento del equipo.

Previo al problema de interés, dimos un breve paseo por otro problema similar conocido como "*The Rest Mismatch Problem*". Para dicho problema vimos algunos resultados teóricos y una heurística que puede ser útil para ciertos deportes donde se encuentra un fixture con 0 o 4 diferencias en los descansos dependiendo de la cantidad de equipos.

Todo esto fue utilizado y aplicado en el torneo de la primera división del fútbol argentino. Este torneo es atípico (con respecto al resto de los países del mundo) en cuanto a la cantidad de equipos que posee. El campeonato sigue el formato de un *single Round Robin* con 26 equipos en el caso del año 2021 y con 28 equipos en 2022. Estos torneos cuentan con una gran cantidad de fechas a jugarse en la segunda mitad del año, sumado a la Copa Argentina que todos juegan y las copas internacionales donde participan al menos 8 equipos más. Todo esto genera que el tiempo para jugar las fechas se vea reducido y sea complicado organizar partidos

entre semana. Adicionalmente, el año 2022 cuenta con el mundial de fútbol a partir del 18 de Noviembre, haciendo aún más difícil la situación.

Dejamos en manos de los organizadores el armado del fixture y los días en los que se debe jugar cada fecha (también puede resolverse con programación matemática) y nos enfocamos en la asignación de partidos a días. Mostramos en la sección de resultados como el uso del modelo mejora los descansos de los equipos reduciendo el total de diferencias que hay entre los equipos. Las mejoras rondaron el 58 % y el 56 % para los años 2021 y 2022, respectivamente, respetando todas las restricciones pedidas.

Más allá de que la mejora del resultado es muy buena, se debe remarcar que existen ciertas restricciones adicionales a las que no tenemos acceso que pueden modificar el resultado aumentando la función objetivo. Este es el caso de algunos partidos fijos en un determinado día o restricciones televisivas.

Otro problema que puede surgir, es que la última fecha se quiera superponer partidos por la definición del campeonato, los descensos o clasificación a copas, pero esto es fácilmente resoluble: Se puede modificar el modelo para asignar los partidos de todas las fechas hasta la ante-última y definir la última de acuerdo a los criterios anteriores.

En principio, este problema fue poco estudiado por los investigadores que se dedican al Sport Scheduling. En el marco teórico del problema hay muchos resultados que pueden obtenerse para generalizar ciertos problemas. Conseguir resultados que, dado el fixture y los días en los que se juega cada fecha, determinen la cantidad de diferencias en los descansos.

En cuanto a los puntos más prácticos, siguiendo con los resultados del modelo, pueden obtenerse fixtures que emparejen más los promedios de descansos de los equipos, haciéndolo más equitativo. En relación a esto, en el trabajo se limitaron las diferencias a un día, salvo por aquellas fechas que van de fin de semana a fin de semana que se les permitió dos días. Una variante del modelo podría ser distribuir de manera equitativa en los equipos las diferencias de dos días antes nombradas, con el fin de que no recaigan en el mismo equipo, tanto a favor como en contra.

Por otro lado, en caso de haber diferencias inevitables, se podría fijar cierto peso en aquellos equipos que viajan más que su rival para que descansen un poco más.

Puede ser interesante estudiar el problema de generar el fixture con la asignación de días en conjunto (recordemos que el fixture ya viene dado para este problema), sabiendo que la complejidad computacional aumentaría considerablemente.

Capítulo 6

Anexo

6.1. Días de descanso - LPF 2022

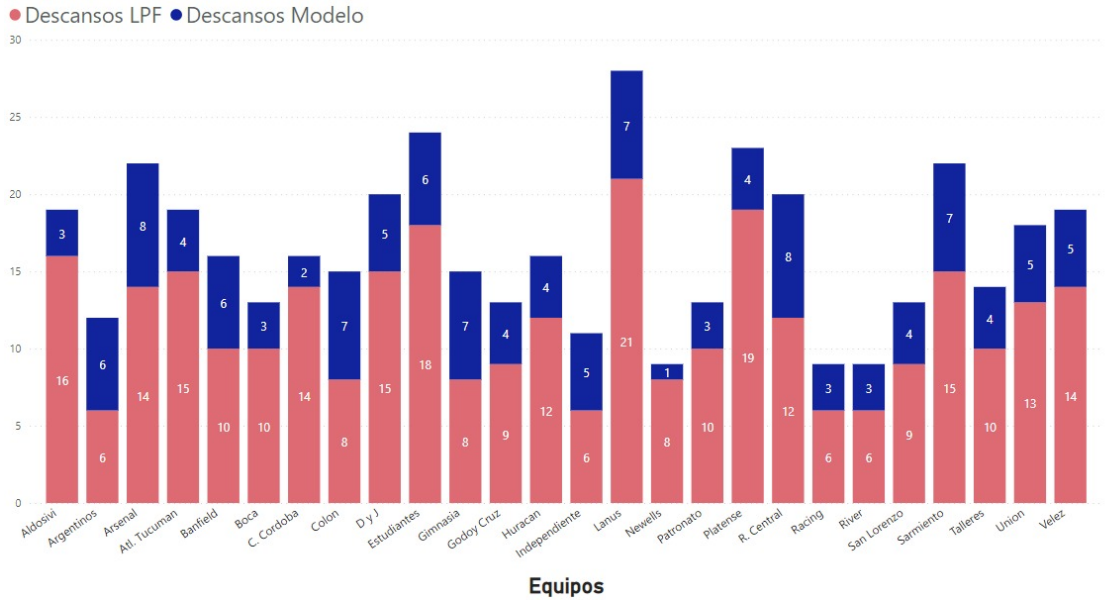
Fecha	Fecha 1	Fecha 2	Fecha 3	Fecha 4	Fecha 5	Fecha 6	Fecha 7	Fecha 8	Fecha 9	Fecha 10	Fecha 11	Fecha 12	Fecha 13	Fecha 14	Fecha 15
Aldosivi	0	4	4	7	5	8	7	4	4	5	7	7	4	7	5
Argentinos	0	5	5	4	6	5	10	5	3	6	5	7	8	4	5
Arsenal	0	6	3	4	8	8	5	6	4	7	6	5	5	6	4
Atl. Tucumán	0	7	4	4	8	5	8	5	4	5	6	7	5	6	6
Banfield	0	7	4	4	5	7	8	8	4	4	5	6	7	6	4
Barracas C	0	8	4	4	5	9	6	7	4	6	5	6	7	5	5
Boca	0	7	3	4	5	7	8	7	3	5	7	6	8	3	4
C. Córdoba	0	9	3	6	4	6	9	4	5	4	9	7	3	5	5
Colon	0	8	3	4	6	7	9	5	3	4	9	6	4	5	4
DyJ	0	5	4	6	5	7	9	4	5	5	7	7	5	5	3
Estudiantes	0	5	4	6	5	9	6	7	3	4	6	8	7	4	4
Gimnasia	0	5	6	4	4	8	8	6	5	3	6	6	9	4	5
Godoy Cruz	0	8	4	4	7	5	8	7	3	5	4	7	6	5	5
Huracan	0	6	4	5	6	8	7	7	3	5	5	8	5	4	4
Independiente	0	6	6	4	7	7	6	6	4	4	8	6	6	3	5
Lanus	0	4	5	4	6	8	7	7	3	4	7	5	8	4	4
Newells	0	5	5	6	5	9	5	7	5	4	5	8	6	4	3
Patronato	0	6	4	4	9	7	5	5	6	6	5	5	7	6	5
Platense	0	8	4	5	5	8	5	6	4	5	7	6	6	4	6
Racing	0	8	4	4	6	7	7	6	3	4	8	6	8	4	4
River	0	6	4	4	6	8	7	7	4	3	7	7	6	4	4
R. Central	0	4	6	4	4	10	4	8	5	5	6	7	5	4	5
San Lorenzo	0	5	5	7	5	7	6	8	3	5	5	8	5	4	6
Sarmiento	0	6	3	4	8	7	5	7	4	4	8	5	7	4	6
Talleres	0	5	4	4	7	7	7	7	5	4	4	8	8	4	4
Tigre	0	6	4	4	7	6	9	6	3	4	7	8	5	4	3
Union	0	7	3	4	5	9	6	8	3	5	4	8	8	4	4
Velez	0	8	4	4	5	7	9	6	3	5	4	8	8	4	4

Cuadro 6.1: LPF 2022- Días de descanso - Fechas 1 a 14

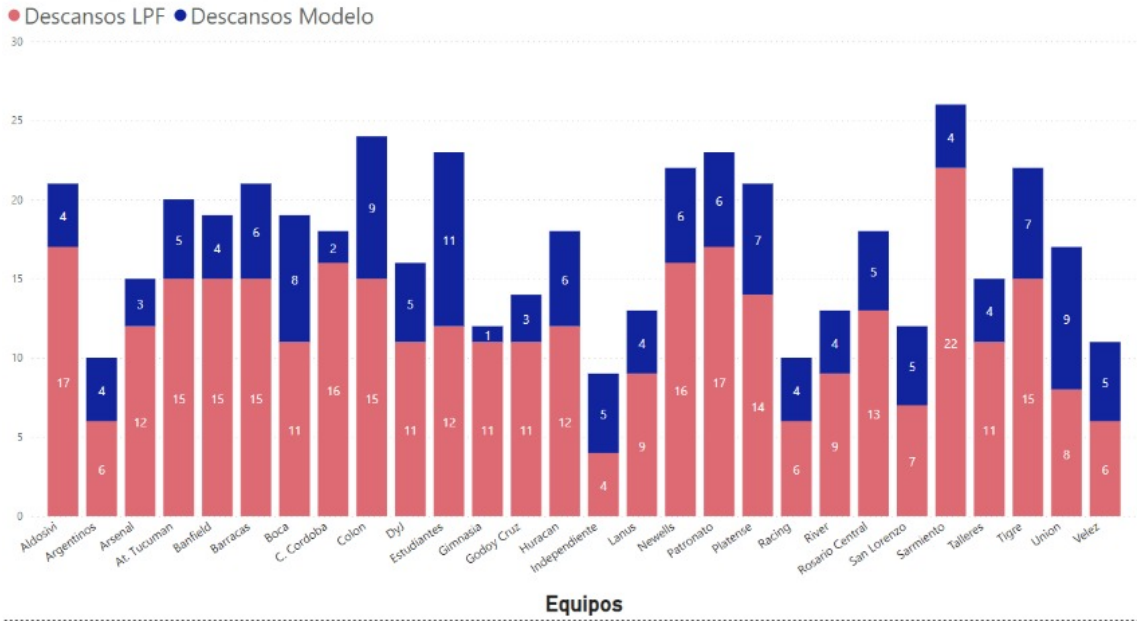
Fecha	Fecha 16	Fecha 17	Fecha 18	Fecha 19	Fecha 20	Fecha 21	Fecha 22	Fecha 23	Fecha 24	Fecha 25	Fecha 26	Fecha 27	Mínimo	Máximo	Promedio
Aldosivi	5	6	5	6	5	7	4	4	5	3	4	6	3	8	5.1111
Argentinos	4	7	5	7	4	7	6	4	4	3	4	5	3	10	5.1111
Arsenal	5	10	6	4	4	5	7	3	4	6	3	7	3	10	5.2222
Atl. Tucuman	5	8	5	5	4	6	7	3	5	3	4	5	3	8	5.1852
Banfield	5	9	5	4	4	5	8	3	5	3	4	7	3	9	5.2222
Barracas C	5	7	5	4	4	8	4	5	4	4	5	7	4	9	5.2963
Boca	7	7	7	3	5	4	9	4	3	3	4	7	3	9	5.1852
C. Cordoba	5	6	9	5	5	6	5	4	5	4	4	4	3	9	5.2222
Colon	8	7	6	4	4	8	4	5	3	6	4	6	3	9	5.2593
DyJ	6	9	6	4	4	5	7	4	5	4	4	4	3	9	5.1482
Estudiantes	6	7	5	6	4	6	5	5	5	4	3	5	3	9	5.1482
Gimnasia	5	6	8	4	5	5	4	7	3	3	5	6	3	9	5.1852
Godoy Cruz	6	9	5	3	4	6	8	3	5	4	3	6	3	9	5.1852
Huracan	6	6	8	4	6	4	8	5	4	4	4	5	3	8	5.2222
Independiente	7	6	5	4	7	5	7	4	4	3	4	7	3	8	5.2222
Lanus	5	8	6	5	4	6	8	4	4	3	5	7	3	8	5.2222
Newells	7	7	8	3	3	7	6	4	6	3	3	8	3	9	5.2593
Patronato	5	7	5	4	7	4	8	3	4	4	3	7	3	9	5.2222
Platense	5	8	5	4	5	7	6	4	3	4	6	7	3	8	5.2963
Racing	5	7	6	4	5	7	5	5	5	4	4	5	3	8	5.2222
River	6	8	7	3	4	6	8	3	4	3	4	7	3	8	5.1852
R. Central	5	7	5	7	5	5	5	3	7	3	3	8	3	10	5.1852
San Lorenzo	5	9	5	4	4	6	7	3	4	4	5	5	3	9	5.1852
Sarmiento	6	6	5	4	5	7	7	3	4	4	5	6	3	8	5.1852
Talleres	5	7	8	4	3	6	8	4	3	4	3	7	3	8	5.1852
Tigre	7	7	6	4	5	7	5	4	5	4	5	6	3	9	5.2222
Union	6	7	4	5	6	6	5	3	5	6	3	5	3	9	5.1482
Velez	6	6	8	4	3	7	7	2	4	5	4	5	2	9	5.1852

Cuadro 6.2: LPF 2022- Días de descanso - Fechas 15 a 27

6.2. Descansos por equipo 2021



6.3. Descansos por equipo 2022



Bibliografía

[1] Ailsa H. Land and Alison G. Doig (1960) "An automatic method of solving discrete programming problems." Econometrica. Vol. 28, no. 3. pp. 497–520.

[2] Atan T, Çavdaroglu B (2018). "Minimization of rest mismatches in round robin tournaments". Computers & Operations Research 99:78–89.

[3] Blest D., Fitzgerald D. (1988). "Scheduling sports competitions with a given distribution of times". Discret. Appl. Math. 22 (1), 9-19.

[4] Briskorn D., Drexl A. (2009). "IP Models for round robin tournaments". Comput. Oper. Res. 36(3), 837-852.

[5] Cavdaroglu B, Atan T (2020) "Determining matchdays in sports league schedules to minimize rest differences". Operations Research Letters 48 (3), 209–216.

[6] Dantzig, George B. (1987) "Origins of the Simplex Method". A History of Scientific Computing, 141–151.

[7] de Werra D. (1981). "Scheduling in sports". In: Hanse, P.(Ed.), Studies on Graphs and Discrete Programming. North-Holland, pp. 381-395.

[8] Durán G., Guajardo M., Sauré D (2017) "Scheduling the South American qualifiers to the 2018" FIFA World Cup by integer programming. Eur. J. Oper. Res. 262(3):1109–1115.

[9] Durán, G., Guajardo, M., Zamorano, G.(2022). "Mathematical models for rescheduling Ecuador's 2020 professional football league season disrupted by COVID-19".

[10] Easton K., Nemhauser G., Trick M. (2001). "The Traveling Tournament Problem: description and benchmarks". In: Walsh T. (Ed.), Principles and Practice of Constraint Programming, Lecture Notes in Computer Science. 2239. Springer, pp. 580-584.

[11] Goossens D., Spieksma F.C.(2012). "Soccer schedules in Europe: an overview". J. Sched. 15 (5), 641-651.

[12] Goossens D., Kempeneers J., Koning H.R., Spieksma F.C. (2015). "Winning in straight sets helps in grand slam tennis". Int. J. Perform. Anal. Sport 15 (3), 1007-1021.

[13] Haselgrove J., Leech J. (1977). "A tournament design problem". Am. Math. Month. 84 (3), 198-201.

[14] Karmarkar, N. (1984) "A new polynomial-time algorithm for linear programming." Combinatorica 4, 373–395.

[15] Klee, V. and Minty, G.J. (1972) "How Good Is the Simplex Method." Shisha, O., Ed., Inequalities III, Academic Press, New York, 159-175.

[16] Knust S. (2008). "Scheduling sports tournaments on a single court minimizing waiting times". Oper. Res. Lett. 36 (4), 471-476.

[17] Knust S. (2010). "Scheduling non-proffesional table-tennis leagues." Eur. J. Oper. Res. 200 (2), 358-367.

[18] Lambrechts E., Ficker A.M., Goossens D., Spieksma F.C.(2016). "Round Robin tournaments generated by the circle method have maximum carry-over". In: Proceedings of the International Conference on Integer Programming and Combinatorial Optimization. Springer, pp. 178-189.

[19] Rey, Pablo A., Comunicación personal.

[20] Rincon F., Soto S (2017) "Introducción a la teoría de polítopos". Lecturas Matemáticas, Volumen 40 (2)(2019), páginas 177-215.

[21] Sanchez J., Sanchez M., Hernandez D., Ramirez-Campillo R., Martínez C., Nakamura F.Y. (2017). "Fatigue in U12 soccer-7 players during repeated one-day tournament games - a pilot study". Journal of Strength and Conditioning Research: November 2019 - Volume 33 - Issue 11 - p 3092-3097.

[22] Schellenberg P., Van Rees G., Vanstone S. (1977). "The existence of balanced tournament designs". Ars Comb. 3, 303-318.

[23] Suksompong W. (2016). "Scheduling asynchronous Round Robin tournaments". Oper. Res. Lett. 44 (1), 96-100.

[24] Tuffaha T, Cavdaroglu B, Atan T (2022) "Round-Robin scheduling with regard to rest differences" TOP. 10.1007/s11750-022-00637-1.

[25] van't Hof P., Post G., Briskorn D. (2010). "Constructing fair round robin torunaments with a minimun number of breaks". Oper. Res. Lett. 38 (6), 592-596.

[26] Wallis W. (1997). "One-Factorizations". Kluwer Academic Publishers.

[27] https://www.ligaprofesional.ar/torneo-primera-lpf/