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

Estudio de la existencia de fixtures con sistema grand-prix para cantidades impares de equipos

Pablo Colombo Villegas

Director: Javier Marenco

Fecha de Presentación
Julio de 2015

Índice general

1. Introducción **9**
- 1.1. Contexto . . . 9
  - 1.1.1. Caso Testigo . . . 10
- 1.2. Existencia de un fixture con sistema Grand-Prix . . . 10
- 1.3. Fixture con sistema Grand-Prix de distancia mínima . . . 14
- 1.4. Resumen de la tesis . . . 15

2. Modelos de programación lineal entera para EGP **17**
- 2.1. Modelo 1 . . . 17
- 2.2. Modelo 2 . . . 18
- 2.3. Comparación de los modelos . . . 21
  - 2.3.1. Comparación de los tamaños de los Modelos 1 y 2 . . . 21
  - 2.3.2. Resultados Computacionales . . . 22

3. Una heurística para MGP **27**
- 3.1. Modelo 3 . . . 27
- 3.2. Tests con el Modelo 3 . . . 29
- 3.3. Enfoque alternativo . . . 30
  - 3.3.1. Búsqueda local . . . 30
  - 3.3.2. Nuestra heurística . . . 31
- 3.4. Resultados de la heurística . . . 32

4. Multiplicación de fixtures **35**
- 4.1. Combinación de m fixtures para n equipos. . . . 36
  - 4.1.1. Separación de equipos y fechas en grupos . . . 36
  - 4.1.2. Cuadrados Latinos . . . 37
  - 4.1.3. Armado de los grupos de fechas . . . 38
  - 4.1.4. Desarmado de partidos rally . . . 39
  - 4.1.5. Verificación de factibilidad . . . 41
  - 4.1.6. Propiedades del fixture construido . . . 42
- 4.2. Consecuencias del teorema de multiplicación de fixtures . . . 43

5. Conclusiones y trabajo futuro **45**

Abstract

Abordamos en este trabajo un formato especial de campeonato para deportes de gimnasio donde la cantidad de equipos es un número impar. En el torneo cada par equipos se enfrenta cuatro veces, dos de ellas con cada localía. Sea n la cantidad equipos. Las fechas están agrupadas de a pares (por ejemplo, se juega los viernes y domingos) y cada par de fechas se denomina un weekend. En cada *weekend*, los equipos se agrupan en $\frac{(n-3)}{2}$ parejas de equipos, y quedan tres equipos sin formar pareja. Cada pareja juega dos partidos entre sí, siendo local uno de los equipos de la pareja. En cuanto a la terna restante de equipos, se selecciona un equipo que será local y recibirá a los otros dos equipos, jugando contra uno de ellos el viernes y contra el otro el domingo. En cada weekend cada equipo juega uno o dos partidos, y son necesarios $2n$ weekends para completar todos los partidos.

El objetivo de este formato es lograr que todos los equipos jueguen todos los weekends, garantizando así la continuidad de su actividad deportiva. Esto no es un problema cuando la cantidad n de equipos es par. Sin embargo, si n es impar, entonces el formato habitual de dejar libre un equipo por weekend conspira contra la mencionada continuidad.

En este trabajo estamos interesados en determinar si existen fixtures con estas características y, en caso de existir, si se pueden solicitar restricciones adicionales sobre el fixture (como por ejemplo, que ningún equipo juegue más de cierta cantidad de partidos seguidos con la misma localía). Por medio de técnicas de programación lineal entera, mostramos que existen fixtures para $n \leq 13$. Este resultado mejora los trabajos anteriores, que habían probado la existencia de fixtures para $n \leq 9$. Además, por medio de técnicas de programación lineal entera, estudiamos la posibilidad de solicitar condiciones adicionales a los fixtures. Finalmente, probamos que si existe un fixture para n equipos y m es impar, entonces existe un fixture para $n.m$ equipos.

In this work we focus in indoor sports with tournaments that have an odd amount of teams where each pair of teams play with each other four times, each team playing two home matches. Let n be the total amount of teams. Two games are held each weekend, for example, on a friday and a sunday. For each weekend, the n teams are separated into $\frac{(n-3)}{2}$ couples and three teams are taken aside. Each couple plays two matches, one of these is home team in both matches. The remaining threesome has one local team who plays to the other two, one on friday and the other one on sunday. Each weekend every team plays one or two matches, it is necessary to play for $2n$ weekends to complete the tournament.

The goal of this kind of tournament is to make every team play every weekend, ensuring the continuity of the sporting activity. This is not a problem when n is even. However, if n

is odd, it is usual to leave a team aside every weekend which conspires with the mentioned continuity.

We are interested in knowing about the existence of these kind of tournaments. And for the n where they exist, we may try and answer the same question with some extra conditions (as an example, no teams playing three consecutive weekends as a visitor). Through integer linear programming, we showed the existance for $n \leq 13$. This extends previous results in the area, which showed existence for $n \leq 9$. On the other hand, through integer linear programming techniques, we studied extra conditions to these tournaments. Finally, we proved that if we have a tournament for n teams and m is odd, then there exists a tournament for $n.m$ teams.

Capítulo 1

Introducción

1.1. Contexto

Esta tesis se enmarca dentro de una disciplina llamada *sports scheduling* que se encarga de estudiar técnicas para diseñar campeonatos deportivos en forma eficiente. Diseñar el torneo implica decir qué equipos juegan entre sí cada fecha y quien será el local. Dado un formato de torneo particular, veremos más adelante que a veces es sencillo armar el torneo y otras veces no. Por ejemplo en el fútbol argentino, hasta el 2014 había 20 equipos y en un torneo de 19 fechas jugaban todos contra todos. Preparar el fixture no es difícil. El problema se vuelve más complejo cuando además ponemos condiciones extra, por ejemplo: los dos equipos de Rosario y Avellaneda no deben ser locales en la misma fecha. Otro ejemplo es exigir que los equipos no sean visitantes tres veces seguidas. Entonces, saber si existe un fixture es la primer pregunta que uno puede hacerse y por cada condición extra, aparece otra pregunta similar. Así, las preguntas se pueden combinar con el objetivo de lograr un torneo acorde con exigencias de la televisión, de la seguridad y también para hacerlo más equitativo.

Una vez que se sabe que existe un fixture, surgen a continuación otras preguntas dentro de Sports Scheduling. ¿Cuál es el mejor fixture de todos?. ¿Cómo elegir el mejor? Se pueden fijar varios objetivos que establecen distintos criterios. Un primer objetivo podría ser minimizar la distancia viajada por los equipos. Por ejemplo, que un equipo de Buenos Aires viaje de Jujuy a Chubut, después a Misiones y finalmente a Santa Cruz pondría en desventaja a este equipo. Probablemente, sea mejor aprovechar el viaje al norte para jugar algún partido más antes de volver.

También se puede establecer un valor para cada viaje de una ciudad a otra, este valor puede ser un promedio ponderado entre el tiempo o la distancia de viaje, el costo, etc. El criterio mencionado concretamente da una función objetivo, esto es una función que evalua cuan bueno es el fixture.

Si el objetivo es que los equipos no juegen de forma consecutiva de local o visitantes, dado un fixture podemos contar la cantidad de veces que un equipo cambia su localía. En este caso, se buscaría el fixture cuya función objetivo dé un valor máximo.

El problema es entonces elegir entre todos los fixtures, el de menor o mayor función objetivo, dependiendo del problema. Este problema tiene dos grandes dificultades. La primera es que muchas veces la función objetivo tiene muchos extremos locales, y por

lo tanto no hay un camino evidente a través de las soluciones factibles que nos lleve al extremo global. La otra dificultad es que el conjunto de soluciones factibles es muy grande, a pesar de ser discreto y finito, su cardinal es muy grande.

1.1.1. Caso Testigo

Hay un caso paradigmático de la situación recientemente mencionada, se llama Traveling Tournament Problem (TTP), que fue estudiado y definido por K. Easton, G. Nemhauser y M.A. Trick en 2001 en [5]. En este trabajo, los autores estudian un formato de torneo sencillo, donde se sabe que existen fixtures factibles. El formato de torneo se llama *double round robin*, round robin significa que juegan todos contra todos, por lo tanto, double round robin significa que juegan todos contra todos de local y visitante. Un ejemplo de este tipo de torneo es la liga de fútbol española.

El objetivo en el TTP es el mismo que en el problema que trataremos en esta tesis, una vez conseguido el fixture factible, elegir el mejor. El criterio usado para elegir el mejor es el mismo, minimizar distancias y respetar un tope para partidos consecutivos como local o visitante.

El problema se define de la siguiente forma:

Input: Un conjunto $T = \{t_1, t_2, \ldots, t_n\}$ con los equipos, una matriz $D \in \mathbb{Z}^{n \times n}$ simétrica donde d_{ij} representa la distancia de la ciudad del equipo i a la del equipo j y por último dos constantes $l \leq u$.

Output: El torneo para los equipos de T donde se minimiza la distancia total recorrida para los equipos y la cantidad máxima de partidos consecutivos como visitante está entre l y u.

No está probado todavía pero se asume que el problema es *NP*-hard. Si se pudo probar en 2009 que sacando la restricción de localías, si es un problema *NP*-hard, la prueba es de Rishiraj Bhattacharyya en **??**. Computacionalmente aparenta ser mucho más difícil que el Traveling Salesman Problem. Hay ejemplos de grupos de equipos con sus costos de viaje y se ha intentado encontrar el mejor fixture según cada cantidad de equipos. Se puede ver en [6] que diez es la máxima cantidad de equipos en un torneo para el cual se conoce el fixture óptimo.

1.2. Existencia de un fixture con sistema Grand-Prix

El problema a resolver en este trabajo consiste en conseguir un fixture para un torneo cuyas características lo hacen especial. El torneo se divide en dos fases. En la primer fase los equipos juegan todos contra todos. Suele ser una fase de consolidación para los equipos y además es clasificatoria para la siguiente. En la segunda, hay llaves y eliminación directa (final, semifinal, cuartos de final, octavos de final, etc). Nosotros nos dedicaremos solamente a la primera. Éste nuevo formato para la primer fase del torneo fue pensando especialmente para la liga Argentina de voley femenino.

Dado que la cantidad de equipos es a veces pequeña y que por diversas razones (por ejemplo, económica) puede variar de año a año, se necesita saber cómo armar un fixture para distintas cantidades de equipos. Consecuencia de ser tan pocos equipos, en la primera fase todos los equipos jugarán entre sí cuatro veces, habrá partidos los viernes y los domingos.

1	2	1	6	1	5	1	4	1	3
6	3	5	2	4	6	3	5	2	4
5	4	4	3	3	2	2	6	6	5

Cuadro 1.1: Fixture para seis equipos.

Si la cantidad de equipos es par, hay un método sencillo para armar un fixture, basta con seguir el esquema de el Cuadro 1.1. En ella pueden verse 5 pares de columnas. Cada par de columnas representa una fecha del torneo. La primer fecha será entonces así: El equipo 1 recibe al equipo 2, el 6 recibe al 3 y el 5 recibe al 4. Se lee de la figura mirando por filas, donde la primer columna tiene al local y la segunda al visitante. Éstas serían las primeras cinco fechas del torneo, que tiene diez, entonces, las últimas cinco pueden ser iguales que las primeras pero invirtiendo la localía. Cada vez que un equipo recibe a otro, juegan entre sí el viernes y el domingo.

¿Por qué decíamos que es sencillo armar este fixture? La razón es que la forma de armarlo es simple, el ejemplo fue para seis equipos pero será igual de fácil para cualquier cantidad par. Se puede ver en todos los pares de columnas, el 1 en el extremo superior izquierdo. Luego, en la fecha 1 están los equipos de 2 a 6 ordenados (recorriendo las columnas en sentido horario) y en las fechas siguientes, se van girando, manteniendo ese orden.

El formato del torneo propuesto para una cantidad n impar se llama Fixture con sistema de Grand-Prix consiste en que los equipos serán separados en parejas (dejando de lado una terna) y para cada par, alguno será el local y jugarán entre sí los dos partidos del fin de semana. Por otra parte, la terna restante de equipos tendrá uno que será el local y recibirá a los otros dos equipos jugando contra uno el viernes y contra el otro el domingo. Conseguimos así que todos los equipos jueguen dos partidos por fin de semana salvo sólo dos, que en vez de quedar libres, juegan un solo partido.

El partido donde un equipo recibe a otros dos será llamado *partido rally*. El equipo local en este partido rally será llamado *equipo rally local* y a los dos equipos que lo visitan los llamaremos *equipos rally visitantes*. La cantidad de fechas en este tipo de fixture es $2n$. Llamaremos EGP al problema de encontrar un fixture con estas características.

Al ser este un formato de torneo nuevo, no se sabe si existen fixtures que cumplan estas condiciones. Por eso es de interés resolverlo. En el año 2009, F. Bonomo, G. Durán y J. Marenco estudiaron el problema EGP y muestran en [1] sus resultados. En el próximo capítulo mostraremos el modelo de programación lineal entera que usaron para intentar resolver el problema, lo llamaremos *Modelo 1*. Con ese modelado del problema encontraron fixtures factibles para cinco, siete y nueve equipos. Más adelante en ese capítulo nosotros propondremos un modelo nuevo para este mismo problema, al cual nos referiremos como *Modelo 2*.

En la Sección 2.3 compararemos ambos modelos y mostraremos los resultados obtenidos luego de una serie de ensayos.

Ejemplo 1.1. En el Cuadro 1.2 mostramos un fixture factible para cinco equipos. Bajo el título de partido regular, se encuentra el partido no rally, ahí hay dos sub columnas, una corresponde a L que tiene al equipo local y otra V que tiene al visitante. Bajo el título partido rally tenemos tres sub columnas, la primera para el equipo rally local y las otras dos para los equipos rally visitantes. Cada fila representa una fecha distinta.

Fecha	Partido regular		Partido rally		
	L	V	L	V_1	V_2
1	5	4	2	1	3
2	5	1	4	2	3
3	3	5	1	2	4
4	1	5	2	3	4
5	5	2	4	1	3
6	3	4	2	1	5
7	1	3	2	4	5
8	3	1	4	2	5
9	3	2	4	1	5
10	5	3	1	2	4

Cuadro 1.2: Fixture factible para cinco equipos.

Se puede obervar en el Cuadro 1.3 que el equipo 5 es rally visitante desde la sexta fecha hasta la novena. Esto quiere decir que en 4 semanas juega sólo 4 partidos, mientras que el el equipo 4 no es visitante más de 2 veces seguidas.

Equipo	Fechas									
	1	2	3	4	5	6	7	8	9	10
1	V_r	V	L	L	V_r	V_r	L	V	V_r	L
2	L	V_r	V_r	L	V	L	L	V_r	V	V_r
3	V_r	V_r	L	V_r	V_r	L	V	L	L	V
4	V	L	V_r	V_r	L	V	V_r	L	L	V_r
5	L	L	V	V	L	V_r	V_r	V_r	V_r	L

Cuadro 1.3: Esquema de localías para el fixture del Cuadro 1.2. Los equipos que tiene una r como subíndice a lado de la V de visitante significa que en esa fecha son equipos rally visitantes.

Esto le significará una diferencia a los equipos.

Teniendo en cuenta este ejemplo, podemos agregarle algunas variantes al problema EGP, éstas pueden ser identificadas con las siguientes preguntas: ¿Se puede armar un fixture tal que...

- ningún equipo juegue más de dos veces seguidas de local o visitante?
- ningún equipo juegue más de j veces seguidas de local o visitante?
- todos los equipos sean igual cantidad de veces equipo rally local?
- ningún equipo es más de i veces rally local?

Todas estas condiciones hacen que el fixture sea más equitativo, por eso tiene interés responderlas e intentaremos hacerlo modelándolas en el Capítulo 2. Veamos ahora otro ejemplo que respeta algunas condiciones extra, y es por lo tanto, más justo.

Ejemplo 1.2. En el Cuadro 1.4, vemos un fixture para siete equipos con condiciones extra, aquí ningún equipo juega más de tres fechas seguidas como local o visitante.

Fecha	Partido regular 1		Partido regular 2		Partido rally		
	L	V	L	V	L	V_1	V_2
1	3	1	5	7	4	2	6
2	1	7	2	5	6	3	4
3	3	6	7	5	4	1	2
4	1	3	5	6	2	4	7
5	2	3	7	4	6	1	5
6	1	2	3	5	4	6	7
7	5	4	7	2	6	1	3
8	2	1	7	6	4	3	5
9	1	6	3	2	4	5	7
10	5	1	7	3	6	2	4
11	1	5	3	4	2	6	7
12	5	2	6	7	4	1	3
13	5	3	7	1	2	4	6
14	1	4	3	7	6	2	5

Cuadro 1.4: Fixture factible para siete equipos con condiciones extra.

En el Cuadro 1.5, mostramos a lo largo del torneo, las localías de cada equipo. Ahí se puede ver cómo para ningún equipo hay tres V o L seguidas. Lo que sí se puede ver es que los únicos equipos que les toca ser rally visitantes en fines de semana consecutivos son el 3 y el 5. Esto sería tal vez una desventaja para ellos y hace que el fixture tampoco sea totalmente equitativo.

Equipo	Fechas													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	V	L	V_r	L	V_r	L	V_r	V	L	V	L	V_r	V	L
2	V_r	L	V_r	L	L	V	V	L	V	V_r	L	V	L	V_r
3	L	V_r	L	V	V	L	V_r	V_r	L	V	L	V_r	V	L
4	L	V_r	L	V_r	V	L	V	L	L	V_r	V	L	V_r	V
5	L	V	V	L	V_r	V	L	V_r	V_r	L	V	L	L	V_r
6	V_r	L	V	V	L	V_r	L	V	V	L	V_r	L	V_r	L
7	V	V	L	V_r	L	V_r	L	L	V_r	L	V_r	V	L	V

Cuadro 1.5: Esquema de localías para el fixture del Cuadro 1.4. Los equipos que tiene una r como subíndice a lado de la V de visitante significa que en esa fecha son equipos rally visitantes.

1.3. Fixture con sistema Grand-Prix de distancia mínima

Una vez resuelto el problema de la existencia de los fixtures para cierta cantidad de equipos, surge la segunda pregunta mencionada al final del capítulo anterior. Supongamos que tenemos como dato los costos del transporte para el viaje para ir de una ciudad a otra, también se podría saber el tiempo de viaje, la distancia recorrida u otras magnitudes. Todas estas cantidades pueden ser englobadas en una nueva que llamaremos *costo*. Entonces tendríamos que el costo de ir de una ciudad a otra sería representado por un número positivo. Con toda esta información disponible, se podrá calcular cuanto debe costarle a cada equipo participar del torneo. Para calcularlo, se toma el fixture y se arma el recorrido que tiene que hacer cada equipo, desde que empieza el torneo en su ciudad, hasta que termina el torneo y tiene que volver. Cada tramo del viaje entre fecha y fecha tiene el costo calculado, y eso se va sumando hasta conseguir la cifra total para el torneo.

Sabiendo eso, podemos intentar contestar la siguiente pregunta: ¿Cuál es el fixture que hace que todos los equipos tengan el menor costo posible a lo largo del torneo?

Este problema es mucho más difícil que el EGP, tiene el nombre de Fixture con sistema de Grand-Prix de distancia mínima (MGP). Haciendo un pequeño paralelo con lo mencionado sobre TTP en el comienzo del capítulo: sabemos que si n es par no es difícil conseguir fixtures factibles ya que el método que dimos nos da muchos. En cambio, sí es difícil conseguir el que minimice las distancias de viaje. En el problema MGP tenemos que resolver una minimización similar pero partiendo de una base distinta, ya no es sencillo conseguir los fixtures factibles.

Esta última versión del problema será abordada en el Capítulo 3. En él mostraremos como modelarlo y la falta de resultados con los *solvers* de programación lineal entera. Entonces presentaremos una solución alternativa que será tomar una solución factible del modelo y mejorarla mediante una heurística.

1.4. Resumen de la tesis

En el Capítulo 1 fue introducido el problema que estudiaremos en este trabajo. Su primera versión con sus subproblemas (que vienen dados por las condiciones extra que se pueden agregar) y su segunda versión.

En el Capítulo 2 veremos cómo fue modelado este problema usando programación lineal entera en el pasado y mostraremos un nuevo modelo. También, compararemos ambos modelos para intentar decir *a priori* cual nos dará mejores resultados. Finalmente, presentaremos los resultados computacionales de los ensayos hechos.

En el Capítulo 3 modelamos con programación lineal entera para MGP y ante imposibilidad de conseguir soluciones con técnicas basadas en PLE, trabajamos el problema con una heurística. En el final del capítulo mostramos los resultados del algoritmo.

Finalmente, en forma paralela al trabajo de encontrar soluciones para los valores de n pequeños, intentamos en este trabajo probar de forma teórica la existencia de fixtures para valores de n arbitrarios. En el Capítulo 4, daremos una condición suficiente para armar un fixture de cualquier tamaño y además mostraremos cómo hacerlo.

En el Capítulo 5 presentamos algunas conclusiones y perpectivas para trabajos futuros.

Capítulo 2

Modelos de programación lineal entera para EGP

2.1. Modelo 1

Existe un modelo para la primera versión del problema, éste fue propuesto en [1]. Si es n la cantidad de equipos (será un número natural), se define $P = \{1, 2, \ldots, n\}$ el conjunto de equipos y $F = \{1, 2, \ldots, 2n\}$ el conjunto de fechas del torneo.

En este modelo, hay que definir tres juegos de variables que son los siguientes: Para cada $i, j \in P$ distintos y para cada $f \in F$ se define la variable binaria x_{ijf} tal que $x_{ijf} = 1$ si y solo si el equipo i recibe *al menos* en un partido al equipo j durante la fecha f. También para cada $i, j \in P$ distintos y para cada $f \in F$ se define la variable binaria w_{ijf} de modo que $w_{ijf} = 1$ si y solo si el equipo i recibe a j en la fecha f y juegan dos partidos, es decir, no juegan en un partido rally. Finalmente, para cada $i \in P$ y $f \in F$ se define la variable binaria y_{if} tal que $y_{if} = 1$ si y solo si el equipo i es el rally local en la fecha f.

Con todas estas variables, la siguiente formulación de programación lineal entera modela el problema de encontrar un fixture para n equipos (donde n es impar).

$$\sum_{f \in F} (x_{ijf} + w_{ijf}) = 2 \qquad \forall\, i, j \in P\ (i \neq j) \tag{2.1}$$

$$\sum_{j \neq i} (x_{ijf} + x_{jif}) = 1 + y_{if} \qquad \forall\, i \in P,\ \forall\, f \in F \tag{2.2}$$

$$\sum_{i \in P} y_{if} = 1 \qquad \forall\, f \in F \tag{2.3}$$

$$w_{ijf} \leq x_{ijf} \qquad \forall\, i, j \in P\ (i \neq j),\ \forall\, f \in F \tag{2.4}$$

$$w_{ijf} \leq 1 - y_{if} \qquad \forall\, i, j \in P\ (i \neq j),\ \forall\, f \in F \tag{2.5}$$

$$x_{jif} \leq 1 - y_{if} \qquad \forall\, i, j \in P\ (i \neq j),\ \forall\, f \in F \tag{2.6}$$

$$x_{ijf},\ w_{ijf} \in \{0, 1\} \qquad \forall\, i, j \in P\ (i \neq j),\ \forall\, f \in F \tag{2.7}$$

$$y_{if} \in \{0, 1\} \qquad \forall\, i \in P,\ \forall\, f \in F \tag{2.8}$$

La ecuación (2.1) asegura que cada equipo juega de local dos veces contra cada uno de

los otros equipos, la ecuación (2.2) dice que cada equipo juega contra sólo un equipo en cada fin de semana a menos que sea el equipo rally local. La ecuación (2.3) garantiza que exactamente un equipo es rally local en cada fecha. Finalmente, el resto de las condiciones fuerza a la correcta relación entre las variables: (2.4) asegura que un partido es jugado dos veces sólo si es jugado al menos una vez, (2.5) dice que el equipo rally local no juega dos partidos contra el mismo equipo en un fin de semana y la ecuación (2.6) garantiza que el equipo rally local es efectivamente local.

Es importante destacar que el modelo con las ecuaciones (2.1)-(2.8) no necesita función objetivo, y al ser un problema de factibilidad, no es fácil de resolver con los algoritmos disponibles basados en *branch-and-bound*, por lo tanto, modificaremos el modelo. Los autores proponen modificar el modelo por uno de optimización, es decir, con función objetivo.

La formulación alternativa se reemplaza la ecuación (2.1) por

$$\sum_{f \in F} (x_{ijf} + w_{ijf}) \quad \leq \quad 2 \qquad \forall\, i, j \in P\ (i \neq j) \tag{2.9}$$

y también se reemplaza la ecuación (2.2) por las siguientes:

$$\sum_{j \neq i} (x_{ijf} + x_{jif}) \quad \leq \quad 1 + y_{if} \qquad \forall\, i \in P,\ \forall\, f \in F \tag{2.10}$$

$$\sum_{j \neq i} (x_{ijf} + x_{jif}) \quad \geq \quad 2y_{if} \qquad \forall\, i \in P,\ \forall\, f \in F \tag{2.11}$$

Por último, introducen la siguiente función objetivo:

$$\text{máx} \sum_{i \in P} \sum_{j \neq i} \sum_{f \in F} w_{ijf}$$

En esta formulación alternativa, cada equipo puede jugar hasta dos partidos por fin de semana, exceptuando el equipo rally local que jugará siempre exactamente dos partidos. Notar que la condición (2.11) es necesaria para evitar fixtures no deseados. Por ejemplo, un fixture que deja libre a un equipo por fecha sería factible en el modelo (2.3)-(2.10).

La función objetivo suma todos los partidos no-rally que hay en el torneo. Sabemos que hay $\frac{(n-3)}{2}$ por fecha y que hay $2n$ fechas. Por lo tanto en el caso de obtener un fixture factible, el valor de la función objetivo debe ser: $n(n-3)$.

Por otra parte, si la función objetivo toma el valor $n(n-3)$, la solución factible donde se alcanza ese máximo debe ser un fixture válido. Las variables y_{if} obligan a los partidos no-rally a estar correctamente distribuidos en las fechas.

2.2. Modelo 2

Proponemos ahora un nuevo modelo para EGP, al igual que en el modelo anterior, se define $P = \{1, 2, \ldots, n\}$ el conjunto de equipos y $F = \{1, 2, \ldots, 2n\}$ el conjunto de fechas del torneo. El conjunto de variables binarias x está indexado sobre i, j, k, f donde $i, j, k \in P$ y $f \in F$.

Interpretaremos las variables de la siguiente forma: $x_{ijkf} = 1 \Leftrightarrow$ el equipo i recibe a los equipos j y k en la fecha f, es importante notar que j y k pueden ser iguales. Para evitar problemas con simetrías, pondremos como condición $j \leq k$. También serán condiciones: $i \neq j$ e $i \neq k$ para que cada equipo reciba a otro equipo distinto.

Si el equipo 1 recibe al 2 y al 3 en la fecha 4, entonces esperamos que $x_{1234} = 1$ en la solución del modelo, podríamos pedir también $x_{1324} = 1$, pero $3 > 2$ entonces esta variable fue eliminada para evitar problemas de simetría. Por otra parte, si el equipo 1 recibe al equipo 2 para jugar 2 veces entre sí en la fecha 3, esperaremos que $x_{1223} = 1$.

Con estas variables, las siguientes ecuaciones modelan el problema de encontrar un fixture para un torneo con n equipos.

$$\sum_{i \in P} \sum_{j \in P} x_{ijjf} = \frac{n-3}{2} \qquad \forall f \in F \tag{2.12}$$

$$\sum_{i \in P} \sum_{j \in P} \sum_{k \in P/j<k} x_{ijkf} = 1 \qquad \forall f \in F \tag{2.13}$$

$$\sum_{f \in F} x_{ijjf} + \sum_{f \in F} \sum_{k \in P/j<k} x_{ijkf} + \sum_{f \in F} \sum_{k \in P/k<j} x_{ikjf} = 2 \qquad \forall i, j \in P / i \neq j \tag{2.14}$$

$$\sum_{j \in P} \sum_{k \in P/j \neq i} (x_{ijkf} + x_{kijf} + x_{jkif}) = 1 \qquad \forall i \in P \ \forall f \in F \tag{2.15}$$

$$x_{ijkf} \in \{0,1\} \qquad \forall i, j, k \in P \ \forall f \in F \tag{2.16}$$

La ecución (2.12) asegura que la cantidad de partidos "normales"(es decir, no-rally) por fecha son $\frac{n-3}{2}$. Con la ecuación (2.13) conseguimos que el partido extra de esa fecha sea efectivamente rally. La ecuación (2.14) asegura que el equipo i juega exactamente dos partidos de local contra el equipo j a lo largo del torneo. Por último, la ecuación (2.15) dice que cada equipo interviene en un solo partido por fecha.

Con este modelo, tenemos el siguiente resultado:

Observación 2.1. Sea $n \in \mathbb{N}$ impar, entonces: existe un fixture para un torneo de n equipos $\Leftrightarrow$ el modelo con las ecuaciones (2.12)-(2.15) es factible.

Nuevamente modelamos EGP como problema de factibilidad. Como dijimos antes, éste no es adecuado para tratar con los algoritmos basados en *branch-and-bound*. Por lo tanto, lo modificamos para conseguir una formulación alternativa.

Para la formulación alternativa reemplazamos la ecuación (2.12) por la siguiente:

$$\forall f \in F : \sum_{i,j \in P} x_{ijjf} \leq \frac{n-3}{2} \tag{2.17}$$

Se puede observar que el cambio se da en la relajación de una restricción. Esto quiere decir que si el modelo original es factible, entonces el alternativo de optimización también. Mas aún, al ser factibles las restricciones, el problema de optimización tendrá óptimo.

Finalmente introducimos la función objetivo:

$$\text{máx} \sum_{i \in P} \sum_{j \in P} \sum_{f \in F} x_{ijjf} \tag{2.18}$$

Con la relajación de la igualdad (2.12) en una desigualdad, alguna de las variables x_{ijjf} podrán ser ceros haciendo a la solución no factible. Para ello se elije tal función objetivo, para forzar a las variables a tomar el valor uno hasta que la suma 2.17 valga $\frac{n-3}{2}$ en todas las fechas del torneo.

Para contestar alguna de las preguntas extra, presentadas en la Sección 1.2, se pueden agregar algunas restricciones más, en este trabajo nos dedicamos a las mismas que en [1] para poder comparar los resultados de ambos modelos. Algunas de ellas no las estudiamos porque para pequeñas cantidades de equipos, no son factibles.

Presentamos las distintas condiciones extra con sus respectivas restricciones para el modelo:

1. Cada equipo es rally local exactamente 2 veces:.

$$\sum_{j \in P} \sum_{k \in P/j<k} \sum_{f \in F} x_{ijkf} = 2 \ \ \forall i \in P$$

2. Cada equipo es rally local a lo sumo 3 veces.

$$\sum_{j \in P} \sum_{k \in P/j<k} \sum_{f \in F} x_{ijkf} \leq 3 \ \ \forall i \in P$$

3. Ningún equipo es rally visitante 2 fechas seguidas.

$$\begin{aligned} \sum_{j \in P/j<i} \sum_{k \in P} (x_{k,j,i,f} + x_{k,j,i,f+1}) \ \ &+ \\ \sum_{j \in P/j>i} \sum_{k \in P} (x_{k,i,j,f} + x_{k,i,j,f+1}) \ \ &\leq \ \ 3 \ \ \forall i \in P \quad \forall f \in F/f \neq 2n \end{aligned}$$

4. Ningún equipo juega de visitante 3 fechas seguidas (igual a la restricción anterior cambiando el 2 final por un 3 y agregando el término correspondiente).

$$\begin{aligned} \sum_{j \in P} \sum_{k \in P/k \leq i} (x_{j,k,i,f} + x_{j,k,i,f+1} + x_{j,k,i,f+2}) \ \ &+ \\ \sum_{j \in P} \sum_{k \in P/k>i} (x_{j,i,k,f} + x_{j,i,k,f+1} + x_{j,i,k,f+2}) \ \ &\leq \ \ 3 \\ \forall i \in P \ \ \forall f \in F/f \leq 2(n-1) & \end{aligned} \tag{2.19}$$

5. Ningún equipo juega de visitante 4 fechas seguidas (también igual a las 2 anteriores poniendo el 4 donde corresponde y agregando los términos correspondientes).

6. Ningún equipo juega de local o visitante 3 fechas seguidas. Uso la condición (2.19) y agrego la siguiente ecuación que limita la cantidad de partidos de local consecutivos:

$$\sum_{j \in P} \sum_{k \in P / j \leq k} (x_{i,j,k,f} + x_{i,j,k,f+1} + x_{i,j,k,f+2}) \leq 2 \quad \forall i \in P \ \forall f \in F$$

7. Ningún equipo juega de local o visitante 4 fechas seguidas (igual al caso anterior, cambiando los 3 por 4).

Esta formulación deberá verificar lo siguiente: Si el óptimo del modelo alternativo es una solución factible con valor $n(n-3) \Leftrightarrow$ esa solución lo es también del modelo original de factibilidad.

Si una solución tiene valor $n(n-3)$, entonces vale la igualdad en las inecuaciones (2.17) pues si para algún f_0:

$$\sum_{i,j \in P} x_{ijjf} < \frac{n-3}{2} \Rightarrow \sum_{i \in P} \sum_{j \in P} \sum_{f \in F} x_{ijjf} < n(n-3)$$

Recíprocamente, si tenemos una solución factible del modelo original, se verifica

$$\sum_{i \in P} \sum_{j \in P} x_{ijjf} = \frac{n-3}{2} \quad \forall f \in F \Rightarrow \sum_{i \in P} \sum_{j \in P} \sum_{f \in F} x_{ijjf} = n(n-3)$$

y por lo tanto tiene que ser el óptimo del modelo alternativo.

2.3. Comparación de los modelos

En esta sección compararemos la cantidad de restricciones y la cantidad de variables para ambos modelos. Primero mostraremos la cantidad de restricciones y variables de cada modelo según la cantidad de equipos que participen en el torneo. Después mostraremos los valores para n entre 5 y 13. Por último, intentaremos sacar alguna conclusión *a priori* sobre cual modelo pordía comportarse mejor en el *solver*.

2.3.1. Comparación de los tamaños de los Modelos 1 y 2

Observando el Cuadro 2.1, podemos ver que el solver genera una cantidad de restricciones de orden $O(n^3)$, mientras que en el Cuadro 2.2 dice que el Modelo 2 tienen menos restricciones, la cantidad es de orden $O(n^2)$.

Por otra parte, en cuanto a la cantidad de variables, se puede ver en el Cuadro 2.3 que el Modelo 1 tiene menos variables que el Modelo 2. Precisamente, el Modelo 1 tiene $O(n^3)$ variables mientras que el Modelo 2 tiene $O(n^4)$.

Si se analiza la matriz del sistema de ecuaciones que tiene como solución a las soluciones factibles del problema de programación lineal entera, esta es de tamaño $r \times p$ donde r es la cantidad de ecuaciones y p es la cantidad de variables. Las matrices de ambos modelos tienen una cantidad de coeficientes de orden $O(n^6)$, por lo tanto, no es evidente cual se comportará mejor, cual dará más soluciones factibles ni cual tendrá mejor rendimiento para valores más altos de n.

Tipo de restricciones	Cantidad de restricciones
2.1	$n(n-1)$
2.3	$2n$
2.4	$2(n-1)n^2$
2.5	$2(n-1)n^2$
2.6	$2(n-1)n^2$
2.9	$n(n-1)$
2.10	$2n^2$
2.11	$2n^2$
Totales	$O(n^3)$

Cuadro 2.1: Cantidad de restricciones para cada tipo en el Modelo 1 según la cantidad de equipos.

Tipo de restricciones	Cantidad de restricciones
2.13	$2n$
2.14	$n(n-1)$
2.15	$2n^2$
2.17	$2n$
Totales	$O(n^2)$

Cuadro 2.2: Cantidad de restricciones de cada clase en el Modelo 2 en función de la cantidad de equipos.

Por último, las Figuras 1 y 2 muestran un gráfico que compara la cantidad de variables y restricciones en función de la cantidad de equipos. Se puede ver en ellos que para los valores de n que trataremos en la siguiente sección, las diferencias están marcadas pero no hace que sean cantidades inmanejables por los *solvers* actuales.

El Modelo 2 tiene una ventaja, o al menos pareciera tenerla ya que si quisieramos decir *en la fecha 1, el equipo 1 es rally local y recibe al 2 y al 3* en el modelo nuevo basta con poner $x_{1231} = 1$ mientras que en el modelo anterior necesitamos $y_{11} = 1$ para decir que el equipo 1 es rally local, $x_{121} = x_{131} = 1$ para decir que el equipo 1 juega de local contra los equipos 2 y 3 en la fecha 1 y por último, $w_{121} = w_{131} = 0$ para decir que el equipo 1 es local contra los equipos 2 y 3 pero no juegan 2 partidos en esa fecha.

2.3.2. Resultados Computacionales

En [1] se presentan los resultados del Modelo 1 usando Cplex 9.1 en una computadora Pentium IV con procesador de 2.5 MHz y 1.25 GB de memoria RAM. Para probar el Modelo 2, usamos SCIP 2.0.1 en una computadora con un procesador AMD Sempron de 1.60 GHz, 1.50 GB de RAM. Por esa diferencia entre equipo y software es que no compararemos los tiempos de ejecución, sino más bien, cuando se logra conseguir una solución.

	Variable	Cantidad
Modelo 1	x_{ijf}	$2n^3$
	w_{ijf}	$2n^3$
	y_{if}	$2n^2$
Modelo 2	x_{ijkf}	$n^3(n-1)$

Cuadro 2.3: Cantidad de variables de cada modelo realtivo a la cantidad de equipos n.

La mínima comparación que puede hacerse, basándonos en que SCIP es *aproximadamente* 4 veces más lento (según algunos experimentos hechos con 87 instancias de MIPLIB2010) es que una instancia que termina en 1 hora en las condiciones de [1] terminará en aproximadamente 8 para los ensayos de este trabajo.

En el Cuadro 2.4 mostramos los resultados de [1].

Instance	$n = 5$	$n = 7$	$n = 9$	$n = 11$
Sin condiciones adicional	Factible	Factible	Factible (implicado por †)	?
Cada equipo es rally local exactamente 2 veces	No factible	Factible ‡	Factible †	?
Cada equipo es rally local a lo sumo tres veces	No factible	Factible (implicado por ‡)	Factible (implicado por †)	?
Sin rally-visitante en semanas consecutivas	Factible	?	?	?
Sin 3 semanas consecutivas de visitante	Factible	?	?	?
Sin 3 semanas consecutivas de o visitante	Factible	?	?	?
Sin 4 semanas consecutivas de visitante	Factible ⋆	?	?	?
Sin 4 semanas consecutivas de local o visitante	Factible	?	?	?

Cuadro 2.4: Resultados computacionales del Modelo 1 con desigualdades válidas adicionales (sin ellas solo la instancia ⋆ terminó en menos de una hora). Donde hay un signo de interrogación, el algoritmo no terminó en una hora.

Por otra parte, los resultados del modelo propuesto en este trabajo resultaron un poco mejores en la práctica. Esto no era tan claro *a priori* ya que no sale claramente beneficiado en la comparación de la cantidad de restricciones y la cantidad de variables vista en la sección anterior.

En el Cuadro 2.5 se presentan los resultados del Modelo 2. En él se puede ver cómo el Modelo 2 logró responder muchas preguntas que el Modelo 1 no había podido. Por ejemplo, sobre la factibilidad de fixtures para once equipos. También pudo contestar las preguntas de si existía fixture con las distintas condiciones extra para 7, 9 y 11 equipos. Por último, con el Modelo 2 se pudo encontrar un fixture factible para 13 equipos. En esta instancia, no se pudo saber si existía con condiciones extra, ahí el *solver* no llegó a terminar de resolver ni siquiera en 8 horas.

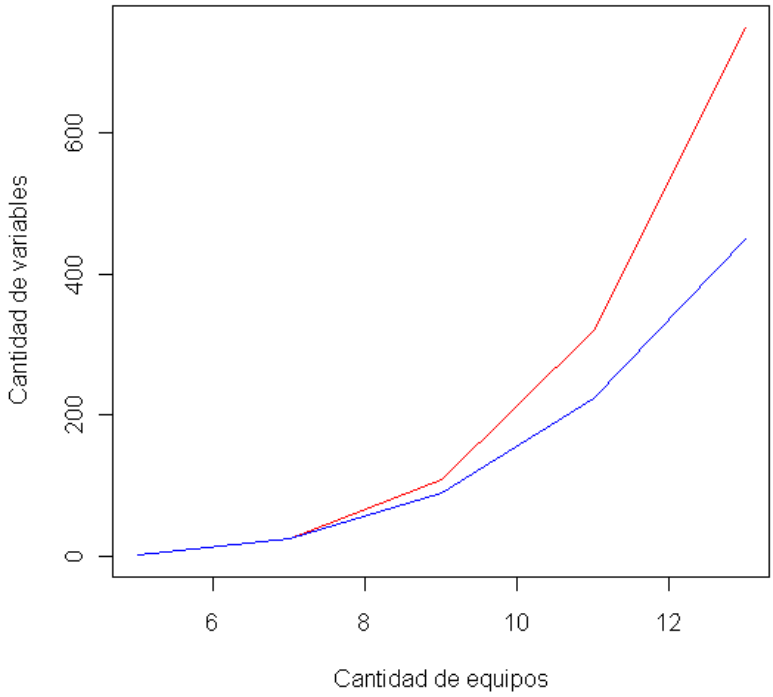

Figura 1: Cantidad de variables en función de la cantidad de equipos. El azul representa al Modelo 1 y el rojo al Modelo 2.

Instancia	$n = 5$	$n = 7$	$n = 9$	$n = 11$	$n = 13$
Sin condiciones adicional	Factible	Factible	Factible (implicado por †)	Factible	Factible
Cada equipo es rally local exactamente 2 veces	No factible	Factible ‡	Factible †	Factible	**?**
Cada equipo es rally local a lo sumo tres veces	No factible	Factible (implicado por ‡)	Factible (implicado por †)	Factible	Factible
Sin semanas consecutivas comoo equipo rally visitante	Factible	Factible	Factible	Factible	**?**
Sin 3 semanas consecutivas de visitante	Factible	Factible	Factible	Factible	**?**
Sin 3 semanas consectuivas de local o visitante	Factible	Factible	Factible	Factible	**?**
Sin 4 semanas consecutivas de visitante	Factible	Factible	Factible	Factible	**?**
Sin 4 semanas consecutivas de local o visitante	Factible	Factible	Factible	Factible	**?**

Cuadro 2.5: Resultados computacionales del *Modelo 2*. Donde las filas que ponen condiciones de localía para 4 fechas consecutivas son factibles como consecuencia de que las filas de 3 también lo son.

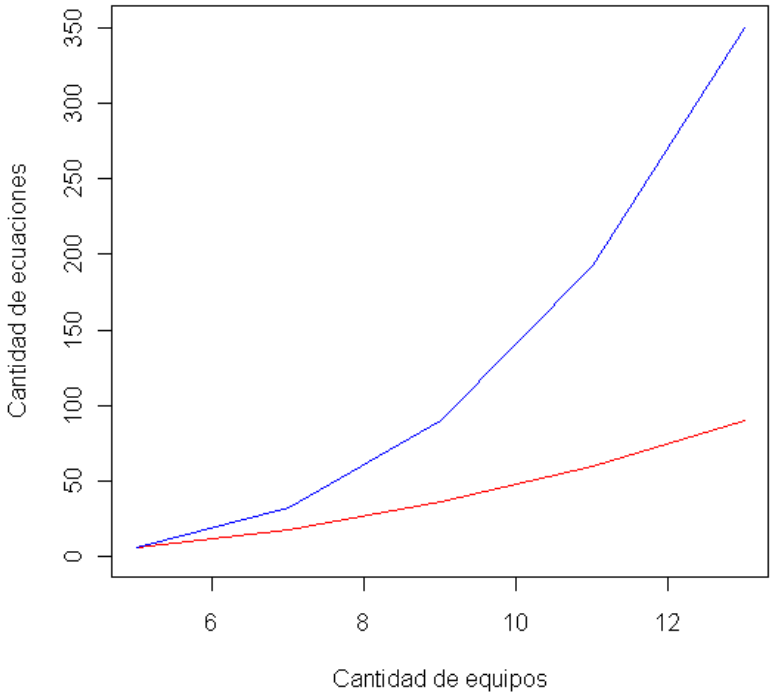

Figura 2: Cantidad de ecuaciones en función de la cantidad de equipos. El azul representa al Modelo 1 y el rojo al Modelo 2.

Capítulo 3

Una heurística para MGP

En este capítulo trataremos el problema MGP. Los datos iniciales serán: una cantidad n de equipos y una matriz C de $n \times n$ que contiene todos números reales no negativos. Esa matriz representará los costos de ir de una ciudad a otra. Por ejemplo, el valor c_{ij} representará el costo de ir de la ciudad donde está el equipo i hasta la ciudad donde está el equipo j, también podrá representar el tiempo de viaje o un promedio ponderado entre ellos. Dada esta interpretación, es claro que en su diagonal, la matriz tendrá todos ceros. Sería razonable además pedir que la matriz sea simétrica, en nuestro caso no lo pediremos porque no es estrictamente necesario.

La matriz con la que nosotros trabajaremos, será tomada de la liga de voley femenino argentino real, y representará simplemente los kilómetros de distancia entre una ciudad y otra, por eso, en nuestro ejemplo será una matriz simétrica.

3.1. Modelo 3

El primer paso para tratar esta nueva versión del problema es adaptar el Modelo 2 de programación lineal entera que teníamos en EGP al problema MGP.

Trabajando en EGP en el Capítulo 2 se tuvo que agregar una función objetivo ficticia para obtener rendimiento óptimo de los *solvers* de programación lineal entera. Ahora, en esta nueva versión del problema, tenemos una función objetivo natural que será una función que sume los costos de todos los equipos.

Quisiéramos que la función objetivo cuente los recorridos de cada equipo. Teniendo una matriz simétrica C tal que c_{ij} representa la distancia de la ciudad donde está el equipo i hasta la ciudad del equipo j. Para conseguir esto, creamos un nuevo juego de variables, dado $i, j, k \in P$ distintos y dado $f \in F_0 = F \cup \{0\} = \{0, 1, \ldots, 2n-1, 2n\}$, la variable z_{ijkf} será binaria y valdrá uno sólo si el equipo i juega en la fecha f en el estadio de j y en la fecha $f+1$ en el estadio de k. La variable con $f = 0$ representa para un equipo el viaje desde su ciudad hasta la ciudad donde juega el primer partido. Análogamente, la variable con $f = 2n$ cuenta el viaje desde la ciudad donde se juega el partido de la última fecha hasta su ciudad.

Es importante notar que, a diferencia de las variables x_{ijkf}, ahora el i, j y k pueden ser iguales o no, no hay restricciones para ellos.

La función objetivo es la siguiente:

$$\text{mín} \sum_{i \in P} \sum_{j \in P} \sum_{k \in P} \sum_{f \in F_0} c_{jk} z_{ijkf} \tag{3.1}$$

Lo último por adaptar del Modelo 2 visto antes sería agregar las restricciones que ligan entre sí a los dos grupos de variables del modelo. Éstas aparecerán después de las restricciones del Modelo 2. Por lo tanto, las restricciones del modelo nuevo, al cual en adelante llamaremos *Modelo 3* serán:

$$\sum_{u \in P/i \leq u} (x_{jiuf} + x_{k,i,u,f+1}) + \sum_{u \in P/i < u} (x_{juif} + x_{k,u,i,f+1}) \leq z_{ijkf} \tag{3.2}$$
$$\forall i,j,k \in P \ / \ i \neq j, \ i \neq k, \ \forall f \in F \ / \ f < 2n$$

$$\sum_{k \in P} \sum_{u \in P} x_{ikuf} + \sum_{u \in P/i \leq u} x_{j,i,u,f+1} + \tag{3.3}$$
$$+ \sum_{u \in P/i < u} x_{j,u,i,f+1} \leq z_{iijf} + 1 \quad \forall i,j \in P \ / \ i \neq j, \ \forall f \in F \ / \ f < 2n$$

$$\sum_{u \in P/i \leq u} x_{j,i,u,f} + \sum_{u \in P/u < i} x_{j,u,i,f} + \tag{3.4}$$
$$+ \sum_{k \in P} \sum_{u \in P/k \leq u} x_{i,k,u,f+1} \leq z_{ijif} + 1 \ \forall i,j \in P \ / \ i \neq j, \ \forall f \in F \ / \ f < 2n$$

$$\sum_{j \in P} \sum_{k \in P} z_{ijkf} = 1 \quad \forall i \in P, \ \forall f \in F \tag{3.5}$$

$$\sum_{k \in P/i \leq k} x_{jik1} + \sum_{k \in P/k < i} x_{jki1} \leq z_{iij0} \quad \forall i,j \in P \tag{3.6}$$

$$\sum_{k \in P/i \leq k} x_{j,i,k,2n} + \sum_{k \in P/k < i} x_{j,k,i,2n} \leq z_{i,j,i,2n} \quad \forall i,j \in P \tag{3.7}$$

El valor de las variables z_{ijkf} es asignado por partes. De acuerdo a la fecha, si es el primer viaje del torneo, si es el último o si es en medio del torneo. Además está separado de acuerdo a si el equipo cambia la localía o no.

La restricción (3.2) le da el valor adecuado a las variables donde el equipo i viaja de la fecha f a la siguiente desde la ciudad donde está el equipo j a la del k. La restricción (3.3) sirve en el caso en que el equipo i es local en la fecha f y visita a j en la fecha $f+1$. La restricción (3.4) es el caso análgo a la anterior, sirve para en el caso en que el equipo i visita a j en la fecha f y es local en la siguiente. La restricción (3.5) ajusta el correcto funcionamiento de las variables nuevas, dice que cada equipo hace un solo viaje entre 2 fechas. Por último, las restricciones (3.6) y (3.7) fijan las variables z_{ijkf} para el viaje al principio y al final del torneo, es decir, para el caso $f = 0$ y $f = 2n$ que no fueron tenidos en cuenta en las restricciones anteriores.

3.2. Tests con el Modelo 3

Como lo habíamos supuesto, el problema MGP es mucho más difícil que el EGP. Buscar un fixture para un torneo de cinco o siete equipos era relativamente rápido, las instancias donde testeamos el modelo en SCIP, no se puede conseguir buenos resultados. A veces es posible encontrar algunas soluciones factibles, pero no podemos saber cuan buenas son ya que la cota superior dada por la relajación lineal es muy mala. Otras veces, para las instancias más grandes, no consigue siquiera una solución factible. Para testear el modelo de programación lineal entera y la heurística, creamos veinte instancias de prueba. Tomando como datos las distancias entre doce equipos argentinos de voley, armamos las primeras cinco instancias tomando cinco equipos al azar entre los doce. Luego, armamos las segundas cinco instancias de la misma forma pero ahora eligiendo siete equipos. Las instancias once a quince son de nueve equipos y por último, de la dieciseis a la veinte son de once equipos.

En el Cuadro 3.1 mostramos el tiempo en que tarda en conseguir la primer solución factible y la cantidad de soluciones factibles al momento de cortar la corrida. Las instancias para cinco equipos corrieron durante treinta minutos, las de siete equipos noventa minutos, las de nueve equipos por ciento ochenta y por último, las de once equipos por seis horas.

Instancia	Tiempo hasta primer solucion	Cant. de soluciones hasta el corte	Gap de optimalidad
1	143	5	503 %
2	124	4	604 %
3	137	2	518 %
4	109	8	591 %
5	133	4	588 %
6	1001	1	1115 %
7	1016	1	1066 %
8	2119	2	930 %
9	935	1	1606 %
10	2160	1	1393 %
11	7039	1	1619 %
12	-	0	-
13	6792	1	1564 %
14	8723	1	2212 %
15	9043	2	2174 %

Cuadro 3.1: Resultados conseguidos en los ensayos de MGP en SCIP.

Como se puede ver en el Cuadro 3.1, SCIP nunca llega a encontrar el óptimo, o al menos, nunca llega a saber si la solución factible encontrada es el óptimo. Las instancias 16 a 20 no están en la tabla, ya que al cabo de seis horas, no encontró solución factible. Tampoco encontró una solución en la instancia 12.

3.3. Enfoque alternativo

Los malos resultados vistos en la sección anterior no fueron una sorpresa. Habíamos mencionado en el primer capítulo que para el TTP la instancia más grande resuelta era de diez equipos. También comentamos las similitudes entre el TTP y el MGP.

Concientes entonces de que conseguir el fixture óptimo, el que menos costos tenga entre todos los fixtures, es una tarea, en la práctica, imposible (por ahora), intentaremos conseguir alguna solución *buena*.

Surge un problema, al no saber el costo del fixture óptimo, no tendremos una forma de medir cuan buena es una solución, entonces lo mediremos en función de cuanto mejoró a partir del fixture con el cual comenzamos.

Hay dos formas de conseguir un fixture cualquiera para comenzar. La mejor es usar los resultados de la sección anterior. Porque aunque no nos haya podido dar el óptimo, tal vez sí pudieron encontrar una solución factible. Si la instancia es tan grande o difícil que no se consiguió ningún fixture factible, podemos usar una solución factible conseguida en el Capítulo 2 usando el Modelo 2. Habrá una diferencia de rendimientos según de donde se sacó el fixture inicial. Si se consiguió intentando minimizar costos tendrá un costo mucho menor que el caso contrario.

¿Qué haremos con esa solución factible? Diseñamos un algoritmo para mejorarla. Ante la diferencia mencionada, podría suponerse que corriendo el algoritmo de mejoramiento de cualquiera de los dos tipos de soluciones (descontando diferencias en tiempo de ejecución) se debería llegar a una solución de calidad similar.

La dificultad está en que la función de costos cuyo dominio son las soluciones factibles no posee un máximo absoluto fácil de encontrar.

3.3.1. Búsqueda local

El algoritmo de mejoramiento que propondremos en la próxima subsección está apoyado en la metaheurística conocida como *Búsqueda local*. La misma está presentada en el libro de B. Korte y J. Vygen [3].

Para estudiar la estructura del algoritmo, necesitamos alguna noción cercanía entre soluciones factibles, que llamaremos *vecindad*. Dada una solución s, las soluciones que estén cerca de ésta serán parte del *vecindario de* s, el cual notaremos $N(s)$.

Para cada par de soluciones factibles s_a y s_b, deben existir otras soluciones factibles $s_1, s_2, \ldots, s_n$ tal que $s_a \sim s_1, s_1 \sim s_2, \ldots, s_n \sim s_b$.

Definimos dos tipos de vecindades distintas:

$N_e(s) = \{s'$ solución factible / s' se obtiene permutando dos equipos de $s\}$

$N_f(s) = \{s'$ solución factible / s' se obtiene permutando dos fechas de $s\}$

Volviendo a la búsqueda local tradicional, presentamos ahora la estructura general del algoritmo:

1. Input: s una solución factible
2. Calcular N(s)
3. Calcular la solución factible s' de N(s) que tiene menor costo.

4. Si s' tiene menor costo que s, poner s=s' y volver a 2.

5. Output: s

3.3.2. Nuestra heurística

Tenemos entonces dos posibles búsquedas locales, una para cada una de las nociones de vecindario: N_e y N_f.

Se presenta un problema, si la función que mide los costos del torneo tiene muchos mínimos locales, entonces la heurística de búsqueda local queda varada en esos mínimos locales y termina muy rápido. En nuestro problema sucede esto mismo, después de aproximadamente diez pasos ambas opciones de búsqueda local se estancan y no llegan a mejorar sustancialmente la solución inicial. No podemos distinguir si alguna de las búsquedas locales funciona mejor que la otra.

Lo que sí notamos es que iterando una detrás de la otra obtenemos mejores resultados. Con esta mejora, el algoritmo igualmente termina en un corto tiempo, entonces, para que no se detenga, hacemos una perturbación aleatoria a la solución factible. Esa permutación logra que la solución factible se aleje de los mínimos locales y explore otra zona de las soluciones factibles. La forma en que se ha hecho es permutando dos equipos o dos fechas sucesivas veces. Es decir, se va de un vecino a otro sin prestar atención a cuanto cambia el costo del fixture. Por esta razón es importante tener guardada la mejor solución a lo largo del algoritmo.

Estas dos ideas ya fueron usadas en [4] y la llamaron *chained local optimization.*

Llamemos BL_e al algoritmo de búsqueda local con noción de vecindarios N_e. Éste recibirá un fixture factible y devolverá una solución factible de costo menor o igual a la que recibió. Análogamente, definimos BL_f a la búsqueda local que usa como vecindario a N_f. Por último llamamos Obj a la función objetivo.

Nuestro algoritmo entonces funcionará así:

1. Input: s una solución factible y un parámetro de corte C

2. Inicializar: $i = 0$ y $s' = s$

3. $s = BL_e(s)$

4. $s = BL_f(s)$

5. Si Obj(s) $\leq$ Obj(s'): s' $= s$

6. s = Perturbación Aleatoria (s)

7. $i = i + 1$

8. Si $i \leq C$, ir a 3.

9. Output: s'.

En los pasos 3 y 4 el algoritmo ejecuta las heurísticas de búsqueda local apoyados en la solución factible s y el resultado de la heurística lo guarda en la misma solución s. Esto

se hace de esta forma porque la solución no puede ser peor que la anterior. La búsqueda local devuelve el input si el en el vecindario no hay una solución mejor.

En el paso 5 se almacena la solución factible de menor costo hasta el momento. En el paso 6, se mezcla aleatoriamente el fixture para que la solución s salga del extremo local donde puede estar encerrada.

3.4. Resultados de la heurística

La primer parte de los ensayos, fue hecha tomando como solución inicial una solución factible del Modelo 2. En la segunda, ingresamos como solución inicial para la heurística a las soluciones factibles del Modelo 3.

El Cuadro 3.2 tiene los resultados de la heurística presentada en la sección anterior. Dado que las soluciones iniciales son malas (pensando en MGP), la heurística tiene la posibilidad de mejorar mucho esas soluciones iniciales. Medimos esa mejora comparando con la solución inicial y tomando el porcentaje. Medimos la capacidad de la heurística de mejorar una solución tomando la mediana de todos los porcentajes, el valor resultante es: 22 %

Presentamos en el Cuadro 3.3 los resultados de la heurística tomando como solución inicial a las dadas por las corridas con el Modelo 3. Se puede ver que la diferencia entre el valor obtenido por la heurística y el valor inicial no es tan marcada como en el cuadro anterior. Eso era de esperar, porque antes la solución inicial de los ensayos no tenía por qué tener un valor de la función objetivo bajo, mientras que ahora esta solución factible fue encontrada intentando minimizar la función objetivo.

Podría pasar que la primer solución factible encontrada usando el Modelo 3 esté muy cerca de ser óptima, por eso no sería raro que la heurística no pueda mejorarla mucho. Por ejemplo en la primer instancia, la heurística sólo puede mejorar la solución en un 5 % su valor. Entonces pueden suceder varias cosas, una opción ese que la solución inicial sea muy buena o tal vez la solución está muy próxima a un máximo local del cual la búsqueda local no consigue escaparse.

Se puede ver también que en la gran mayoría de las instancias, el valor inicial de la heurística del Cuadro 3.2 es mayor que el valor de la solución inicial ingresada para los ensayos del Cuadro 3.3. Hay alguna excepción, por ejemplo la séptima instancia. Esto muestra que la función objetivo tiene muchos máximos locales y que a veces el algoritmo de búsqueda local se queda trabado en ellos. Pero además nos dice que la solución factible que nos da el Modelo 3 es en general buena.

Instancia	Valor inicial		Valor del parámetro de corte 20	30	40	50	Mejora
1	40527	Tiempo:	3	4	5	6	19 %
		Valor:	33663	33663	33454	32872	
2	46721	Tiempo:	2	3	4	6	22 %
		Valor:	36902	36902	37129	37129	
3	38155	Tiempo:	2	3	5	6	29 %
		Valor:	27643	27637	27416	27416	
4	25658	Tiempo:	2	3	4	5	10 %
		Valor:	23318	23338	23318	23318	
5	37641	Tiempo:	2	3	4	6	20 %
		Valor:	30288	31031	30288	30288	
6	69257	Tiempo:	15	19	27	39	26 %
		Valor:	51777	51604	51777	51777	
7	72022	Tiempo:	15	19	28	30	28 %
		Valor:	54185	54286	52061	54286	
8	71864	Tiempo:	14	19	27	33	27 %
		Valor:	53385	53377	52916	53850	
9	63055	Tiempo:	12	20	25	33	27 %
		Valor:	47547	47585	46247	47595	
10	59275	Tiempo:	13	21	29	43	11 %
		Valor:	52997	52997	53864	55315	
11	127580	Tiempo:	49	84	116	136	18 %
		Valor:	107500	105339	104834	105099	
12	111152	Tiempo:	62	95	119	132	22 %
		Valor:	88718	87008	88242	88242	
13	121256	Tiempo:	56	81	110	130	21 %
		Valor:	98279	97483	99177	96609	
14	125674	Tiempo:	60	84	122	150	20 %
		Valor:	101525	103547	101776	101776	
15	124574	Tiempo:	59	90	107	150	22 %
		Valor:	96389	96352	97168	96238	
16	202626	Tiempo:	148	234	280	346	22 %
		Valor:	162908	158460	158592	160675	
17	184548	Tiempo:	159	220	297	372	25 %
		Valor:	139810	139781	139267	139810	
18	199113	Tiempo:	143	217	294	353	25 %
		Valor:	150893	151470	150893	151668	
19	211795	Tiempo:	156	221	266	363	25 %
		Valor:	158983	158999	160708	158983	
20	198136	Tiempo:	548	818	1142	1553	26 %
		Valor:	150709	148398	147755	149579	

Cuadro 3.2: Resultados conseguidos por la heurística usando como fixture inicial el dado por el Modelo 2. Los tiempos están expresados en segundos.

Instancia	Valor inicial		Valor del parámetro de corte 20	30	40	50	Mejora
1	25720	Tiempo:	4	7	10	13	5 %
		Valor:	24579	24579	24579	24579	
2	27197	Tiempo:	4	6	9	11	12 %
		Valor:	24132	24132	24132	24132	
3	21743	Tiempo:	4	7	9	10	20 %
		Valor:	17703	17594	17703	17703	
4	19209	Tiempo:	4	7	8	11	5 %
		Valor:	18421	18421	18421	18421	
5	25291	Tiempo:	4	7	9	11	6 %
		Valor:	24213	23963	24005	23963	
6	59509	Tiempo:	13	19	29	33	28 %
		Valor:	43385	43694	43666	43385	
7	52498	Tiempo:	12	16	26	31	6 %
		Valor:	50097	49593	50303	50303	
8	50389	Tiempo:	12	18	24	30	9 %
		Valor:	45991	45991	45991	45991	
9	47493	Tiempo:	14	17	26	33	17 %
		Valor:	33082	33082	33082	33082	
10	59275	Tiempo:	13	20	28	32	17 %
		Valor:	49406	49975	49406	49406	
11	109084	Tiempo:	45	63	92	116	11 %
		Valor:	98129	98993	97852	97852	
13	102310	Tiempo:	51	79	109	127	13 %
		Valor:	89822	89359	89440	89539	
14	107958	Tiempo:	43	75	91	117	12 %
		Valor:	95493	96058	97283	96476	
15	99142	Tiempo:	44	64	87	108	8 %
		Valor:	91999	91496	91844	91310	

Cuadro 3.3: Resultados conseguidos por la heurística usando como fixture inicial el dado por el Modelo 3. Los tiempos están expresados en segundos.

Capítulo 4

Multiplicación de fixtures

Una vez conseguidos todos los fixtures que el modelo de programación lineal entera nos pudo dar, buscamos la forma de usarlos para armar fixtures para valores de n más grandes.

En este capítulo se demostrará una serie de hechos que nos permitirá probar, al final, que dado un fixture para n equipos, con n impar, podremos conseguir un fixture para $n.m$ equipos siendo m cualquier número impar. A medida que avanzamos con las propiedades que nos darán el resultado esperado, en paralelo iremos mostrando un ejemplo para armar un fixture para 15 equipos partiendo de uno de 5 mostrado en el Cuadro 4.1 más adelante.

Lema 4.1. *Sea n impar, dado que $2n$ es la cantidad de fechas de un torneo entonces cada equipo es rally-visitante exactamente cuatro veces.*

Demostración. Llamamos x a la cantidad de fines de semana en que un equipo juega dos partidos (es decir, no es rally visitante) e y a la cantidad de partidos donde juega un solo partido (es decir, es rally visitante). Entonces se verifica que $x+y = 2n$. Además, como cada equipo juega 4 veces con los otros equipos, cada equipo juega $4(n-1)$ partidos, entonces, tenemos también la ecuación: $2x + y = 4(n-1)$. Resolviendo el sistema, obtenemos que $x = 2n-4$ e $y = 4$. Es decir, cada equipo debe ser rally visitante exactamente 4 veces. □

Los grafos que permiten más de un arista entre sus vértices se llaman *multigrafos*, el multigrafo que mencionaremos a continuación es un caso particular de multigrafo, ya que podrá tener hasta dos aristas entre cada par de vértices.

Dado un equipo A, llamamos $E_1, E_2, \ldots, E_m$ a los equipos que recibe en fines de semana rally. Sean $V = \{E_1, E_2, \ldots, E_m\}$ y $E = \{(i,j) \ / \ i$ y j visitan a A en el mismo fin de semana$\}$.

Entonces $G_A = (V, E)$ es un multigrafo no dirigido. En los siguientes lemas estudiaremos este grafo.

Lema 4.2. *El grafo G_A se puede separar en ciclos.*

Demostración. ¿Cuantas aristas salen de cada vértice E_i? Cada vértice representa un partido rally-visitante. Ahora bien, por cada visita del equipo E_i al equipo A hay una arista, entonces también debe estar su par. La razón está dada porque cada equipo tiene que jugar dos partidos de visitante contra otro. Si en una fecha lo visitó y jugaron un

solo partido, tendrá que visitarlo y jugar otro partido, es decir, lo visitó como equipo rally visitante dos veces.

Al ser par la cantidad de aristas que salen de cada vértice, el multigrafo se llama *Euleriano* y por lo tanto, se puede separar en ciclos. □

Lema 4.3. *Dado un equipo A, éste visita como equipo rally a exactamente dos equipos.*

Demostración. Es claro combinando el hecho de que A juega dos partidos de visitante contra cada equipo y con el hecho de que es equipo rally visitante exactamente cuatro veces a lo largo del torneo. □

Lema 4.4. *Dado un equipo A y un equipo B al que A visita en un fin de semana rally. Entonces A aparece en exactamente dos ciclos de grafos G_B correspondientes a sus visitas como equipo rally visitante a dos equipos distintos.*

Demostración. Este lema es consecuencia de los dos anteriores. Llamemos B y C a los equipos que visita A como rally-visitante. Como A visita a B en un fin de semana rally, entonces pertenece a uno de los ciclos de *los visitantes de B*. Pero además A visita a otro equipo, C, entonces también pertenece al ciclo *los visitantes de C*. No podría pertenecer a otro, ya que significará que es equipo rally visitante más de cuatro veces. □

Definición 4.5. Dada una arista de un grafo, tenemos sus dos vértices. Arbitrariamente vamos a elegir uno de ellos y decir que está marcado mientras que el otro vértice no.

Observación 4.6. Sea C un ciclo del grafo G_A formado por los visitantes de cierto equipo A. Marcando un vértice en cada arista, es posible elegir una vez a cada equipo. La forma de hacerlo es recorriendo el ciclo y eligiendo alternadamente los vértices.

En la Figura 1 se pueden ver los distintos grafos correspondientes a las visitas en partidos rally del torneo para 5 equipos del Cuadro 4.1. Los equipos 1 y 3 son rally-local solo una vez mientras que el equipo 4 es rally-local dos veces. Arbitrariamente se cambiaron las aristas del grafo por flechas para hacer dirigido el grafo y que resulte un ciclo. Cuando tenemos una arista (i, j) en el grafo, eso quiere decir que i y j visitan juntos en un fin de semana para jugar un partido cada uno. En ese caso, el equipo marcado será el i, el equipo de donde sale la flecha.

Si uno de los ciclos tiene solo dos vértices, como es el caso de los visitantes del equipo 4, ponemos un asterísco en alguno de los vértices del grafo. Eso quiere decir que comenzamos a recorrer el ciclo desde el vértice con asterísco. Por ejemplo, el ciclo formado por los equipos 1 y 2 lo leemos primero con la arista $(1, 2)$ y después con la arista $(2, 1)$.

4.1. Combinación de m fixtures para n equipos.

4.1.1. Separación de equipos y fechas en grupos

Iniciamos ahora el proceso para multiplicar los fixtures. Partimos sabiendo que dado cierto n impar, existe un fixture para ese valor. Tomamos otro impar, m y construiremos un fixture para nm equipos. Primero se dividen en m grupos de n equipos:

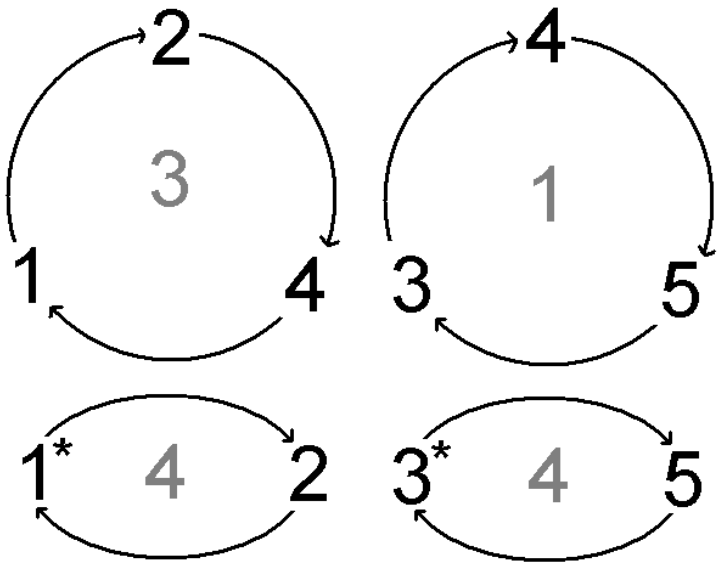

Figura 1: Grafos que resultan de unir a los equipos que visitan juntos como rally-visitantes a un cierto equipo. Cada ciclo tiene en el centro al equipo local. El asterisco indica por que equipo se empieza a recorrer el ciclo.

- Grupo 1: $E_1^1, E_2^1, \ldots, E_n^1$,
- Grupo 2: $E_1^2, E_2^2, \ldots, E_n^2$,
  $\vdots$
- Grupo m: $E_1^m, E_2^m, \ldots, E_n^m$.

También se divide el fixture en m grupos de $2n$ fechas, de la fecha 1 a la $2n$ será el grupo 1, de la fecha $2n+1$ hasta la $4n$ será el grupo 2 y así siguiendo hasta el grupo m que va de la fecha $2(m-1)n+1$ hasta la fecha $2mn$.

Para el ejemplo con $n = 5, m = 3$, los tres grupos de fechas serán los siguientes: fechas 1 a 10, fechas 11 a 20 y fechas 21 a 30. Como el valor de m es pequeño, los tres grupos de equipos serán numerados del 1 al 5 pero se distinguirán por colores. Habrá un grupo de equipos rojo, otro verde y otro azul, representando respectivamente a los grupos de equipos 1, 2 y 3.

4.1.2. Cuadrados Latinos

Supongamos que tenemos una matriz $M \in \{1, 2, \ldots, m\}^{m \times m}$ tal que en cada fila y en cada columna no se repiten números (es decir, están todos los números desde el 1 hasta el m). Estas matrices son conocidas, se llaman *cuadrados latinos*. Se puede ver un ejemplo en la Figura 2.

Además se le exigen a la matriz dos condiciones extra: que los elementos de la diagonal sean todos distintos (podemos suponer que en la posición (k, k) está el número k) y que sea simétrica. Una matriz de estas características nos servirá para armar el fixture para

1	5	2	3	4
5	2	4	1	3
2	4	3	5	1
3	1	5	4	2
4	3	1	2	5

Figura 2: Cuadrado latino de tamaño 5×5.

nm equipos, por eso, debemos mostrar la existencia de ellas para cualquier m natural e impar. Si tomamos un grupo abeliano $(G, *)$ de orden m y armamos la tabla de la operacion $*$, conseguimos una matriz simétrica. Si por ejemplo armamos la tabla para el grupo $(\mathbb{Z}_m, .)$ obtenemos una mariz cuya diagonal tiene los m valores distintos. No verifica que la posicion (k, k) tenga el número k pero para conseguirlo, basta con permutar los números adecuadamente.

La matriz que se usará en el ejemplo para $n = 5$ y $m = 3$ es una matriz de 3×3 que se puede ver en la Figura 3.

1	3	2
3	2	1
2	1	3

Figura 3: Cuadrado latino de tamaño 3×3.

4.1.3. Armado de los grupos de fechas

Teniendo la matriz necesaria y un fixture para n equipos, veamos cómo interpretando la matriz, conseguimos un fixture para nm equipos. Si en la posición (i, j) de la matriz está el número k, entonces en el $k-$ésimo grupo de fechas, jugarán entre sí los equipos del grupo i contra los del j. ¿Cómo lo harán? Mirando el fixture de $n \times n$ conocido, ponemos como locales a los equipos del grupo i y como visitantes a los del grupo j. Como además la matriz es simétrica, en la posición (j, i) también habrá un k, entonces en el mismo grupo de fechas estará el resto de los equipos del grupo j recibiendo al resto de los equipos del grupo i. De esta forma aparecen todos los equipos en cada grupo de fechas.

Tomando como base el fixture del Cuadro 4.1, mostraremos cómo armar el primer grupo de fechas siguiendo el procedimiento explicado en el párrafo anterior.

Trabajaremos primero construyendo el fixture para el primer grupo de fechas. Como se puede ver en el Cuadro 4.2, el primer paso es juntar tres fixtures para $n = 5$ y en él, poner a jugar los equipos de los distintos grupos según el cuadrado latino. Como en el cuadrado hay un 1 en el lugar (1,1), eso quiere decir que en el primer grupo de fechas, los equipos del grupo 1 (en la figura rojo) juegan entre sí. Entonces en el primero de los 3 fixutres que juntamos, ponemos todos equipos del grupo 1 a jugar entre sí.

Fecha	Partido regular		Partido rally		
	L	V	L	V_1	V_2
1	2	3	1	4	5
2	1	2	4	3	5
3	5	1	4	3	5
4	2	1	4	3	5
5	5	2	1	3	4
6	2	5	3	1	4
7	5	3	4	1	2
8	2	4	1	3	5
9	3	5	4	1	2
10	5	4	3	1	2

Cuadro 4.1: Fixture factible para $n = 5$.

Además, hay un uno en la posicion (2,3), eso quiere decir que en el segundo fixture de los tres que juntamos, ponemos como locales a equipos del grupo 2 (azul en la figura) y como visitantes a los del grupo 3 (verde en la figura). Análogamente, como el cuadro latino lo elegimos simétrico, hay un 1 en la posición (3,2), entonces completamos el tercero y último de los fixtures poniendo como locales a los equipos del grupo 3 (verdes) y a los del grupo 2 (azules) como visitantes.

Fecha	Partidos que resultan de juntar tres fixtures de 5 equipos														
	L	V	L	V_1	V_2	L	V	L	V_1	V_2	L	V	L	V_1	V_2
1	2	3	1	4	5	2	3	1	4	5	2	3	1	4	5
2	1	2	4	3	5	1	2	4	3	5	1	2	4	3	5
3	5	1	3	2	4	5	1	3	2	4	5	1	3	2	4
4	2	1	4	3	5	2	1	4	3	5	2	1	4	3	5
5	5	2	1	3	4	5	2	1	3	4	5	2	1	3	4
6	2	5	3	1	4	2	5	3	1	4	2	5	3	1	4
7	5	3	4	1	2	5	3	4	1	2	5	3	4	1	2
8	2	4	1	3	5	2	4	1	3	5	2	4	1	3	5
9	3	5	4	1	2	3	5	4	1	2	3	5	4	1	2
10	5	4	3	1	2	5	4	3	1	2	5	4	3	1	2

Cuadro 4.2: Primer paso: juntamos los tres fixutres de tamaño $n = 5$ y asignamos los grupos de equipos. Éste no es todavía un fixture factible.

4.1.4. Desarmado de partidos rally

Por lo hecho hasta ahora, hay m partidos rally por fecha, y esto no nos sirve para el formato de torneo que pretendemos armar. Por lo tanto, hay $m - 1$ de ellos que desarmaremos. Los elegidos para desarmar serán los partios rally donde intervienen equipos

de distintos grupos, es decir, el único partido rally de la fecha será el que corresponde al número de la diagonal en la matriz, supongamos para fijar ideas que es el 1.

¿Cómo desarmar esos partidos rally? Supongamos para simplificar la notación, que el equipo A recibe a B y C en un partido rally del fixture conocido de $n \times n$. Después de analizar los ciclos del grafo G_A y haber elegido a una vez a cada equipo, se procede a mirar cuál de los dos equipos (B ó C) es el elegido en ese fin de semana que se busca desarmar. Si es B, entonces se saca a B de ese partido y se deja al equipo A recibiendo a C solo, ahí disputarán dos partidos. Cuando se haga esto para todos los partidos rally correspondientes a esa fecha, se tendrá todos los equipos B, salvo uno, ese equipo B faltante es el del primer grupo de equipos. Entonces se tiene a todos los equipos B salvo el 1.

Si en la fecha jugaban los equipos del grupo i contra los del j, entonces el equipo B del grupo i jugará contra el equipo B del grupo j. Como cada equipo es elegido dos veces en cada grupo de fechas, entonces habrá otro momento en donde separaremos a los equipos de B salvo el del grupo 1, ahí podremos hacer que jueguen de nuevo entre sí invirtiendo la localía.

Hemos sacado a B del partido rally donde A recibe a B y C, ¿cuándo recibe entonces A a B en el torneo? Afirmamos antes que de las dos veces que B recibe a A en el torneo de n equipos en una es elegido (y por lo tanto sacado) y en la otra no. Por lo tanto, en la otra vez que visita a A como equipo rally, el otro equipo rally visitante debe ser elegido y por lo tanto sacado y en esa fecha jugarán los dos partidos que corresponde a A recibir al equipo B.

Así se logra armar con los $m-1$ equipos separados de cada fecha, $\frac{m-1}{2}$ partidos nuevos. Por lo tanto, tenemos $m(\frac{n-3}{2}+1)+\frac{m-1}{2}$ partidos por fecha, esto es exactamente $\frac{mn-3}{2}+1$ que es el número necesario.

Además, la cantidad de fechas, poniendo m grupos de $2n$ fechas es $2mn$. También el número esperado.

Siguiendo los ciclos, en el Cuadro 4.3 podemos ver como quedan desarmados los partidos rally. Por ejemplo, para la primer fecha, tenemos que el equipo 1 azul recibe al 4 y 5 verde y el equipo 1 verde recibe al 4 y 5 azul. El grafo que corresponde a los equipos que visitan a 1 está orientado de forma que está el arista (4,5) pero no el arista (5,4). Entonces el equipo elegido será el equipo 4. Sacamos los equipos 4 de los partidos rally para que jueguen entre sí y los equipos número 1 recibirán a los equipos 5 (del color contrario). Hay un caso particular, que se trata un poco distinto. Como el grafo de los equipos que visitan a 4 se divide en dos ciclos disjuntos que tienen sólo dos equipos, entonces, aparece en el grafo el arco (1,2) y el (2,1). La pregunta es entonces: ¿Cuál miramos primero?. Para contestar esta pregunta está el asterisco en el grafo. El equipo 1 tiene un asterisco, eso quiere decir que será elegido primero. Por eso, en las fechas donde el equipo 4 es rally local contra los equipos 1 y 2, el primero en ser elegido es el 1. En la fecha 7 los equipos 1 jugarán entre sí mientras que en la fecha 9 serán los equipos 4.

Una vez hecho esto, podemos reemplazar estos cambios en lo que teníamos hasta ahora y tener terminado el primer grupo de fechas. Eso se puede ver en el Cuadro 4.4.

Repitiendo este proceso para el segundo y tercer grupo de fechas, podemos obtener el fixture completo para 15 equipos, este tendrá 30 fechas. Presentamos la solución final para hacer algunas observaciones. El fixture terminado se puede ver en el Cuadro 4.5

Fecha	Partidos rally							Partidos regulares					
	L	V_1	V_2	L	V_1	V_2		L	V	L	V	L	V
1	1	4	5	1	4	5		1	5	1	5	4	4
2	4	3	5	4	3	5		4	5	4	5	3	3
3	3	2	4	3	2	4		3	4	3	4	2	2
4	4	3	5	4	3	5		4	3	4	3	5	5
5	1	3	4	1	3	4		1	4	1	4	3	3
6	3	1	4	3	1	4	$\Rightarrow$	3	1	3	1	4	4
7	4	1	2	4	1	2		4	2	4	2	1	1
8	1	3	5	1	3	5		1	3	1	3	5	5
9	4	1	2	4	1	2		4	1	4	1	2	2
10	3	1	2	3	1	2		3	2	3	2	1	1

Cuadro 4.3: Como transformar dos partidos rally en tres partidos regulares.

Fecha	Partidos que resultan de juntar tres fixtures de 5 equipos														
	L	V	L	V	L	V	L	V	L	V	L	V	L	V_1	V_2
1	2	3	2	3	2	3	1	5	1	5	4	4	1	4	5
2	1	2	1	2	1	2	4	5	4	5	3	3	4	3	5
3	5	1	5	1	5	1	3	4	3	4	2	2	3	2	4
4	2	1	2	1	2	1	4	3	4	3	5	5	4	3	5
5	5	2	5	2	5	2	1	4	1	4	3	3	1	3	4
6	2	5	2	5	2	5	3	1	3	1	4	4	3	1	4
7	5	3	5	3	5	3	4	2	4	2	1	1	4	1	2
8	2	4	2	4	2	4	1	3	1	3	5	5	1	3	5
9	3	5	3	5	3	5	4	1	4	1	2	2	4	1	2
10	5	4	5	4	5	4	3	2	3	2	1	1	3	1	2

Cuadro 4.4: Fixture terminado para el primer grupo de fechas.

4.1.5. Verificación de factibilidad

Veamos que cada equipo recibe exactamente dos veces a cualquier otro a lo largo del torneo. Sean $E_i{}^j$ y $E_r{}^s$ dos equipos cualquiera. Para analizar separamos en tres casos:

1. $j = s$. Los dos equipos pertenecen al mismo grupo.
2. $j \neq s$ e $i = r$. Los dos equipos son de distintos grupos pero corresponden al mismo índice en el orden dentro del grupo.
3. $j \neq s$ e $i \neq r$. Los dos equipos son de distintos grupos y corresponden a distintos índices en el orden dentro de sus grupos.

Demostramos caso por caso:

1. Supongamos que k es un grupo al que pertenecen los dos equipos. Entonces el k-ésimo grupo de fechas jugarán entre si un sub-torneo los n equipos del grupo k. Como

estamos mirando con quien juegan los equipos del grupo k en el k-ésimo grupo de fechas, los partidos rally no fueron desarmados. Además, por ser ese grupo de fechas un torneo de $n \times n$ válido, cada equipo recibe dos veces a cada uno de los otros.

2. Para simplificar la notación, supongamos que hablamos de los equipos A de los grupos i y j. Si vemos en la matriz, en la posición (i, j) y (j, i) estará el mismo número, supongamos k. Eso quiere decir que separamos los partidos rally del k-ésimo grupo de fechas para estos grupos de equipos. El equipo A fue elegido dos veces en los ciclos donde estaba (una vez por cada ciclo al que pertenece), y por lo explicado antes, habrá dos partidos, uno donde el equipo A del grupo i recibe al del j y juegan dos partidos y otro igual pero invirtiendo la localía.

3. También para simplificar la notación, supongamos que hablamos del equipo A del grupo i y del equipo B del grupo j. Como antes, supongamos que en la posición (i, j) y (j, i) estará el número k. Entonces en el k-ésimo grupo de fechas jugarán entre sí los equipos del grupo i y los del j. Si en el fixture de $n \times n$ con el que comenzamos hay una fecha donde el equipo A recibe al equipo B sin estar en un partido rally, entonces, gracias al número k en la posición (i, j) el equipo A del grupo i recibirá al equipo B del grupo j. Como el fixture es válido, también el equipo B recibe al A en alguna otra fecha, entonces el equipo B del grupo j también recibe al equipo A del grupo i. Si A recibiera a B en un partido rally, entonces, habrá exactamente dos fechas donde A recibe a B en partidos rally. En una de ellas, B será elegido y no jugará con A, sino con otro equipo B de otro grupo. Pero en el fin de semana donde B no es elegido, estos jugarán entre sí dos partidos con A como local.

Con todo lo dicho en este capítulo, hemos demostrado el siguiente resultado:

Teorema 4.7. *Dado un fixture para una cantidad impar n equipos, podemos construir uno para $n.m$ equipos (donde también m es impar).*

4.1.6. Propiedades del fixture construido

¿Cuantas propiedades se conservan cuando *multiplicamos* los fixtures como recién hemos hecho?. El fixture para cinco equipos que usamos en un principio verificaba que ningún equipo jugaba más de tres veces seguidas como local o visitante. Si miramos el que obtuvimos para quince equipos, éste también verifica esa propiedad. Para que así suceda, hay que chequear sólo dos cosas:

1. Los equipos que eran rally visitantes pero desarmamos su partido y fueron elegidos, pasan en el nuevo fixture a ser locales. Entonces eso puede romper con la propiedad de la localía, podría ser cuatro veces seguidas local. Para prevenir esto, cuando armamos el partido con los dos equipos elegidos, elegimos la localía de modo que el que es local no lo haya sido antes o lo sea después tal que en total sean cuatro partidos seguidos de local. En el ejemplo fue posible pero no hay una demostración de que la idea pueda generalizarse.

2. La cantidad de partidos consecutivos como local o visitante es en general menor o igual a tres dentro del fixture que se usa como base, lo que podría suceder (pero

tampoco sucede en el ejemplo) es que un equipo sea local (o visitante) al final de un fixture y al principio del otro, y cuando se *multiplica* ese fixture, este equipo sea local (o visitante) más de tres veces seguidas, donde esas, por lo menos, cuatro fechas estén comprendidas entre dos grupos de fechas distintos.

Quedará para trabajo futuro conocer las condiciones suficientes más débiles para que este tipo de propiedades se conserven para cantidades de equipos más grandes.

4.2. Consecuencias del teorema de multiplicación de fixtures

Con el resultado visto en la sección anterior se pueden sacar por lo menos dos conclusiones útiles, que son las siguientes:

Observación 4.8. Como $15 = 3 \times 5$ y tenemos un fixture para $n = 3$ (ó para $n = 5$), tendremos uno para $n = 15$. Lo mismo para $n = 21, 25, 27, 33, \ldots$. En la Figura 4 resaltamos los valores de n impares menores a 100 para los cuales podremos conseguir un fixture gracias a este teorema. En rojo están los valores para los cuales el modelo de programación lineal entera encontró un fixture y en azul los valores para los cuales el teorema lo ha hecho apoyándose en los resultados obtenidos antes.

		5	7	9	11	13	15	17	19
21	23	25	27	29	31	33	35	37	39
41	43	45	47	49	51	53	55	57	59
61	63	65	67	69	71	73	75	77	79
81	83	85	87	89	91	93	95	97	99

Figura 4: El Teorema 4.7 combinado con los resultados computacionales obtiene soluciones factibles para los números marcados en azul.

Observación 4.9. Dado un numero n impar cualquiera, basta con conseguir un fixture de tamaño p, donde p es algún número primo que divide a n y con eso podemos armar el fixture para n equipos.

Fecha	Partidos que resultan de juntar tres fixtures de 5 equipos														
	L	V	L	V	L	V	L	V	L	V	L	V	L	V_1	V_2
1	2	3	2	3	2	3	1	5	1	5	4	4	1	4	5
2	1	2	1	2	1	2	4	5	4	5	3	3	4	3	5
3	5	1	5	1	5	1	3	4	3	4	2	2	3	2	4
4	2	1	2	1	2	1	4	3	4	3	5	5	4	3	5
5	5	2	5	2	5	2	1	4	1	4	3	3	1	3	4
6	2	5	2	5	2	5	3	1	3	1	4	4	3	1	4
7	5	3	5	3	5	3	4	2	4	2	1	1	4	1	2
8	2	4	2	4	2	4	1	3	1	3	5	5	1	3	5
9	3	5	3	5	3	5	4	1	4	1	2	2	4	1	2
10	5	4	5	4	5	4	3	2	3	2	1	1	3	1	2
11	2	3	2	3	2	3	1	5	1	5	4	4	1	4	5
12	1	2	1	2	1	2	4	5	4	5	3	3	4	3	5
13	5	1	5	1	5	1	3	4	3	4	2	2	3	2	4
14	2	1	2	1	2	1	4	3	4	3	5	5	4	3	5
15	5	2	5	2	5	2	1	4	1	4	3	3	1	3	4
16	2	5	2	5	2	5	3	1	3	1	4	4	3	1	4
17	5	3	5	3	5	3	4	2	4	2	1	1	4	1	2
18	2	4	2	4	2	4	1	3	1	3	5	5	1	3	5
19	3	5	3	5	3	5	4	1	4	1	2	2	4	1	2
20	5	4	5	4	5	4	3	2	3	2	1	1	3	1	2
21	2	3	2	3	2	3	1	5	1	5	4	4	1	4	5
22	1	2	1	2	1	2	4	5	4	5	3	3	4	3	5
23	5	1	5	1	5	1	3	4	3	4	2	2	3	2	4
24	2	1	2	1	2	1	4	3	4	3	5	5	4	3	5
25	5	2	5	2	5	2	1	4	1	4	3	3	1	3	4
26	2	5	2	5	2	5	3	1	3	1	4	4	3	1	4
27	5	3	5	3	5	3	4	2	4	2	1	1	4	1	2
28	2	4	2	4	2	4	1	3	1	3	5	5	1	3	5
29	3	5	3	5	3	5	4	1	4	1	2	2	4	1	2
30	5	4	5	4	5	4	3	2	3	2	1	1	3	1	2

Cuadro 4.5: Fixture terminado para 15 equipos.

Capítulo 5

Conclusiones y trabajo futuro

En los primeros capítulos de este trabajo propusimos un nuevo modelo para EGP. Los resultados fueron muy buenos, conseguimos fixtures factibles para instancias de mayor tamaño que las que se conocían hasta el momento. Además, el modelo nuevo tiene unas variables que permiten expresar las restricciones que vienen del torneo de forma sencilla en las ecuaciones. Queda para trabajo futuro pensar cómo hacer para conseguir fixtures factibles para una cantidad mayor de equipos. Las opciones son por lo menos tres, se podría buscar una nueva forma de modelar el problema o también se podría intentar encontrar familias de desigualdades válidas. También, se podría diseñar una heurística para encontrar soluciones factibles.

Una vez que comenzamos a tratar con MGP, la tarea fue mucho más difícil, ya no era posible conseguir la solucion óptima con el *solver* de programación lineal entera. Entonces, tomamos un camino alternativo. Si el solver no consiguió siquiera soluciones factibles, la heurística permitió conseguir un fixture *bueno*. En cambio, si el solver pudo conseguir alguna solución factible, con la heurística la hemos podido mejorar en un poco y conseguir una solución relativamente buena. Queda para trabajo futuro conseguir por un lado algún buen algoritmo heurístico que pueda mejorar mucho una solución factible cualquiera, incluso las conseguidas por el modelo que no contempla minimizar distancias. Por otro lado, se puede intentar demostrar que la solución factible es efectivamente buena y que no se puede mejorar significativamente.

En el Capítulo 4 conseguimos un resultado que nos da una condición suficiente para la existencia de fixtures. Sirve por ejemplo para saber para qué cantidades de equipos tenemos que enfocarnos en conseguir fixtures factibles según lo dicho más arriba. No necesitamos por ejemplo encontrar un modelo que nos dé una solución factible para 21 equipos, pues este lo conseguimos usando el Teorema 4.7. Lo que queda por contestar en el futuro es: Dado un numero p primo, ¿existe un fixture para p equipos?. No parece una pregunta para nada sencilla. Lo que logramos en este trabajo son soluciones factibles para algunos valores pequeños y una forma de multiplicarlos. No conocemos argumentos teóricos que nos aseguren la existencia de soluciones factibles. Sería un proyecto ambicioso intentar encontrar ese argumento mencionado.

Bibliografía

[1] F. Bonomo, G. Durán, y J. Marenco, *Exploring the existence of fixtures for scheduling sport leagues with odd numbers of teams and grand-prix weekends.* VI ALIO/EURO Workshop on Applied Combinatorial Optimization (2008).

[2] Bhattacharyya, Rishiraj. .A note on complexity of traveling tournament problem..Optimization Online 2480 (2009).

[3] B. Korte y J. Vygen, "Combinatorial optimization: theory and algorithms". Springer-Verlag, 2000.

[4] Martin, Olivier C., and Steve W. Otto. Çombining simulated annealing with local search heuristics..Annals of Operations Research 63.1 (1996): 57-75.

[5] K. Easton, G. Nemhauser, y M.A. Trick, *The Traveling Tournament Problem description and benchmarks.* T. Walsh editor, Principles and Practice of Constraint Programming, Vol 2239 of Lecture Notes in Computer Science, 580-584. Springer, 2001.

[6] http://mat.gsia.cmu.edu/TOURN/