

TESIS DE LICENCIATURA

Aplicación del Travelling Tournament Problem
para el diseño de fixtures deportivos
en torneos por parejas

Alejandro Gabriel Burzyn

LU:253/04

aleburzyn@gmail.com

Director: Dr. Javier Marengo

jmarengo@dc.uba.ar

Co-Directora: Dra. Flavia Bonomo

fbonomo@dc.uba.ar

Co-Director: Dr. Guillermo A. Durán

gduran@dm.uba.ar.

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Septiembre de 2010

Resumen

La planificación eficiente de fixtures es una tarea muy compleja para las entidades organizadoras de los torneos. No sólo es difícil encontrar buenas soluciones, sino que también existe una variedad de requerimientos cambiantes que pueden invalidar a último momento las soluciones ya obtenidas.

En este trabajo se aborda el caso particular del diseño de fixtures para torneos por parejas, donde los equipos se dividen en parejas a lo largo del campeonato. En estos torneos los fixtures se planean tomando a las parejas como si fueran equipos. Un enfrentamiento entre dos parejas dura dos fechas, durante las cuales los equipos de la pareja visitante visitarán alternadamente a los equipos de la pareja local en un orden preestablecido. También se reserva una fecha para que los integrantes de una misma pareja se enfrenten entre sí.

El objetivo de esta tesis es estudiar el problema de diseñar un fixture minimizando la suma de la distancia total que deberán recorrer los equipos durante el torneo. Se toma como punto de partida teórico el Travelling Tournament Problem (TTP) propuesto por M. Trick ([9]). A efectos de obtener fixtures más eficientes, se proponen relajaciones al formato de torneo por parejas, como variar el orden en que se juegan los partidos entre los equipos de dos parejas que se enfrentan, o permitir recombinar las parejas a lo largo del campeonato.

El problema es modelado con distintos modelos de programación lineal entera basados en el TTP. A partir de estos modelos se propone un esquema algorítmico que permite obtener soluciones eficientes tanto en calidad como en tiempo. Este esquema aplica como última etapa una metaheurística Tabu Search, construida a partir de versiones en modelos de programación entera de búsquedas locales usadas comúnmente en el TTP.

Abstract

The efficient planning of fixtures is a very complex task for tournament organizers. Not only it is difficult to find good solutions, but there is also a variety of changing requirements which may invalidate any previously obtained solutions.

This work addresses the case of fixtures for tournaments where the teams are grouped into couples. In such a tournament, the fixtures are planned by taking couples as teams. A match between two couples involves two dates, during which the teams from the visiting couple play against the teams in the local couple in a prespecified order. There is also a special date for playing the matches between the teams of each couple.

The objective of this thesis is to study the problem of designing a fixture minimizing the sum of the total distance travelled by the teams during the tournament. As a theoretical starting point we take the Travelling Tournament Problem (TTP) proposed by M. Trick ([9]). In order to obtain more efficient fixtures, we propose relaxations of the tournament format, such as changing the order in which matches are played between the teams of two couples and allowing the recombination of couples throughout the tournament.

The problem is modeled with different integer linear programming models based on the TTP. A computational procedure is proposed which allows to obtain good solutions in efficient computational times. This procedure applies as a last step a Tabu Search metaheuristic, built from integer programming versions of local search techniques commonly used in the TTP.

Agradecimientos

A Javi, mi director, por su gran predisposición, optimismo e infinita paciencia. Esta tesis no habría sido posible de no ser por él.

A mis codirectores Willy y Flavia, por todas sus colaboraciones.

A mis amigos de la facu: Gabi, Dami, Maty, Jona, Martiniano (y muchos otros más), con quienes he compartido toda esta etapa.

A Guido, Agustín, Mariela, Alex, Maxi y Fede, mis amigos de la vida, que me han acompañado tanto en las buenas como en las malas.

A mis hermanos Pablo y Dalia, que han sido para mí un ejemplo a seguir, y siempre han sabido estar en el momento justo con la palabra justa.

A mi cuñada Lucila, que es como una hermana más para mí.

A mi sobrina Laura, que nació hace poco pero ya la quiero mucho.

A mis padres Mirta y Enrique, que me apoyaron incondicionalmente en todo momento, no habría llegado hasta este punto de no ser por ellos.

Índice general

1. Introducción	7
1.1. Marco Teórico	9
1.1.1. Teoría de grafos	9
1.1.2. Optimización combinatoria	10
1.1.3. Programación lineal entera	11
1.1.4. Metaheurísticas y búsquedas locales	12
1.2. Características de campeonatos y fixtures	13
1.3. El Travelling Tournament Problem (TTP)	14
1.4. Descripción del campeonato de voleibol	16
1.5. Resumen de la tesis	17
2. Modelos propuestos	20
2.1. Definiciones preliminares	20
2.2. Modelo 1	21
2.3. Modelo 2	28
2.4. Modelo 3	32
2.5. Comparación de los modelos	37
2.5.1. Variables	37
2.5.2. Restricciones	38
2.5.3. Efectividad de los modelos	39
3. Estrategias de resolución	40
3.1. Procedimiento original	40
3.2. Nuestra propuesta	43
3.2.1. Definición de la configuración de parejas	43
3.2.2. Modelo de parejas prefijadas y esquema de visita fijo	45
Modelo 1	46
Modelo 2	47
Modelo 3	49
3.2.3. Esquema de visita dinámico: metaheurística en base a programación entera	53
Implementación de las búsquedas locales con programación entera	53
HAP de parejas prefijado	53
Intercambio de rivalidades entre equipos	54

<i>ÍNDICE GENERAL</i>	6
Intercambio de fechas	55
Intercambio de equipos	58
Metaheurística Tabu Search con programación entera	59
4. Resultados	62
5. Conclusiones y trabajo a futuro	70
A. Instancias de prueba	76

Capítulo 1

Introducción

Planificar el fixture para un campeonato deportivo implica definir precisamente qué partidos se jugarán entre qué equipos y en qué fechas. La tarea se vuelve cada vez más compleja mientras mayor sea la escala del campeonato, ya sea en cantidad de equipos (se multiplican los problemas de disponibilidad equipo-fecha), dispersión geográfica de los mismos (se incrementan los costos de viaje), o inclusive la popularidad del mismo campeonato (se desea obtener un fixture atractivo para los espectadores, con partidos importantes en fechas convenientes, a fin de vender entradas e incrementar beneficios con contratos televisivos). Ante estas situaciones, las instituciones organizadoras suelen contentarse con obtener un fixture que cumpla lo indispensable, pero sería ideal obtener un fixture que resuelva lo mejor posible todas las necesidades de los actores involucrados (instituciones, equipos y espectadores).

Tomando esto en cuenta, diseñar el fixture para una liga deportiva es una tarea altamente compleja. Se debe respetar una gran cantidad de requisitos, algunos impuestos por la liga, otros por los mismos equipos. De estos requisitos, algunos deben ser satisfechos completamente para considerar un fixture como válido (por ejemplo, no usar un estadio en una fecha donde no está disponible), los demás deben ser respetados lo mejor posible (como hacer que los equipos viajen lo menos posible durante el campeonato). Estos últimos requisitos definen un criterio de optimización para elegir entre varios fixtures válidos. Todo esto hace que el planeamiento de fixtures sea un problema idóneo para ser tratado con herramientas de optimización combinatoria.

En el ámbito deportivo, es común que surjan nuevos requisitos a último momento antes de la publicación del fixture, por ejemplo, un estadio podría no estar disponible para la fecha que se le asignó. Estas situaciones podrían empeorar la optimalidad del fixture propuesto hasta el momento, o inclusive invalidarlo (como en el caso del ejemplo). En consecuencia, aparte de obtener fixtures de calidad, es importante también conseguirlos en el menor tiempo posible, para que idealmente pueda recalcularse un nuevo fixture inmediatamente ante cualquier imprevisto.

Las características propias de cada liga definen un criterio totalmente particular para diferenciar “buenos” fixtures de “malos” fixtures, por lo que

no se puede pensar en un método universal para resolver este tipo de problemas. Un ejemplo sencillo de esto es que si, por el formato del campeonato, los equipos que juegan de visitante varias veces seguidas vuelven a su ciudad entre juego y juego, no tiene sentido minimizar las distancias recorridas por éstos puesto que en este caso serán constantes (siempre y cuando jueguen todos contra todos de local y visitante). En esta situación, el criterio de optimización suele ser el de crear un fixture donde los juegos de local y visitante de cada equipo estén lo más intercalados posible.

Al problema de obtener un fixture que minimice la distancia total recorrida por los equipos se lo conoce como Travelling Tournament Problem (TTP, [9]). Este problema es muy estudiado en la rama de investigación operativa llamada sports scheduling, y es muy difícil de resolver óptimamente incluso para cantidades pequeñas de equipos.

Algunas ligas, como las de voleibol y básquetbol argentinas, y la de voleibol brasileña, presentan un formato de campeonato menos usual que la mayoría, ya que los fixtures se organizan en base a parejas de equipos. Cuando dos parejas se enfrentan, cada integrante de una pareja deberá jugar contra los dos de la segunda. En este trabajo abordaremos este tipo de fixture. Para modelarlo utilizaremos extensiones y derivaciones de modelos del TTP.

Este trabajo está motivado por la generación de fixtures para la Primera División de la Liga Argentina de Vóleybol Masculino, la cual tiene 12 equipos distribuidos por todo el país, los que a su vez se dividen en 6 parejas fijas durante todo el campeonato. Los partidos se planean por pareja y por weekends. Un weekend está conformado por dos partidos que se juegan Jueves y Sábado de una misma semana. Cuando dos parejas se enfrentan en un weekend, el orden de los partidos estará prefijado en base al ranking de los equipos en el campeonato anterior. En esta liga, cuando un equipo debe jugar varios partidos seguidos de visitante, no regresa a su ciudad entre medio de éstos, sólo lo hace cuando le corresponde jugar de local o se acaba el campeonato. Esta razón, sumada a la distribución geográfica de los equipos, hace que sea de gran interés para los equipos no viajar en exceso.

Antes de armar el fixture se toman en cuenta los intereses/posibilidades de cada equipo de jugar de local o visitante en determinadas fechas. Por ejemplo, un equipo puede no querer jugar de local porque su estadio está en refacciones, pero por otro lado, puede ser que dado un evento como un festival local, al equipo le convenga jugar de local porque tendrá mayor recaudación de entradas.

Actualmente se han obtenido fixtures eficientes para este campeonato utilizando un modelo de programación lineal entera y una metaheurística tabu-search [2], ambos basados en el TTP. Para ello se redujo el problema considerando las parejas como equipos, y los weekends como fechas, lo que resulta en una variante del TTP de 6 equipos.

Las parejas son determinadas por los organizadores del campeonato sin hacer uso de herramientas formales. El criterio seguido es el de asociar equipos cercanos con el fin de disminuir la distancia recorrida. Así, dos equipos

que se encuentren en una misma ciudad probablemente serán pareja, pero eventualmente puede ocurrir que formen parejas equipos muy distantes entre sí, lo que podría significar que la elección no es óptima. Este problema surge y se potencia especialmente en los casos donde los dos equipos de una pareja no pueden jugar de local/visitante a la vez en determinada fecha.

En este trabajo propondremos automatizar la generación del fixture desde la elección de parejas inclusive y, por otro lado, analizaremos la posibilidad de relajar incrementalmente diversos aspectos del sistema del campeonato actual, con el objetivo de mejorar las distancias recorridas por los equipos.

En primer lugar, analizaremos la posibilidad de definir automáticamente las parejas. El siguiente paso será relajar el orden en que se enfrentan los equipos de las parejas, permitiendo que se enfrenten en cualquier orden en cada weekend. Finalmente, en vistas de mejorar las posibles ineficiencias de la elección de parejas, en lugar de definir las 6 parejas a priori y sostenerlas durante todo el campeonato, se relajará esa condición y se exigirá solamente que se respete el formato de pareja vs pareja dentro de cada weekend, permitiendo recombinar las parejas para el siguiente weekend. Esta última modificación debería permitir no sólo evitar los casos poco eficientes de apareo sino que también aprovechará casos no evidentes en que tal vez juntar dos equipos lejanos permita alcanzar un fixture mejor.

Para ello, analizaremos una gama de modelos de programación entera que modelan el problema en su forma más relajada (parejas dinámicas). Utilizaremos versiones restringidas de estos modelos para atacar gradualmente el problema. Asimismo, plantaremos y estudiaremos versiones en modelos de programación entera de los mecanismos de búsqueda local más comúnmente utilizados en las metaheurísticas aplicadas al TTP.

1.1. Marco Teórico

1.1.1. Teoría de grafos

A continuación daremos algunas definiciones básicas sobre grafos.

Un *grafo* G es un par (V, E) donde V es un conjunto de *nodos* (o *vértices*), y $E \subseteq V \times V$ es un conjunto de *arcos* o *aristas* entre los nodos de V .

Cada arista se representa como el par de nodos que une, así (a, b) denota la arista que une los nodos a y b . Dos nodos unidos por una arista se consideran *adyacentes* o *vecinos*.

Si se les agrega direcciones a las aristas, el conjunto E pasará a ser de pares ordenados, (a, b) denotará la arista que va *desde* el nodo a *hacia* el b . En este caso diremos que G es un *digrafo*.

Dado un grafo o digrafo G , se le pueden asignar pesos numéricos tanto a los nodos como a las aristas. En este caso diremos que los nodos o las aristas del grafo son *pesados*.

Llamaremos *camino* a una secuencia ordenada de nodos de G , donde dos

nodos sucesivos en la secuencia son vecinos. Si G es un digrafo, el camino debe ser *dirigido*, es decir, los dos nodos sucesivos deben ser vecinos por una arista que vaya del primero en la secuencia al siguiente. Si el camino comienza y termina en el mismo nodo, diremos que es un *circuito*.

G es un *grafo completo* si están definidas todas las aristas posibles, es decir $E = V \times V$.

1.1.2. Optimización combinatoria

Un problema es de optimización combinatoria cuando posee estas características:

- Un conjunto finito dado C .
- Un conjunto de condiciones que separan al espacio de subconjuntos de C en soluciones válidas (factibles) o inválidas (no factibles).
- Una función objetivo $f : Sub(C) \rightarrow \Re$ que tiene como dominio los subconjuntos de C y les asigna un valor.
- Se pide encontrar de entre todas las soluciones factibles, alguna que maximice (o minimice) f .

Entre los problemas más conocidos y estudiados en Optimización combinatoria, podemos encontrar los siguientes: *Assignment Problem*, *Travelling Salesman Problem (TSP)*, *0-1 Knapsack Problem*, *Set Covering Problem*, etc [25].

La forma más directa de resolver estos problemas es por enumeración de las soluciones factibles, evaluando f una por una y quedándose con la mejor. Lamentablemente este método resulta extremadamente costoso, puesto que la cantidad de soluciones factibles suele ser exponencial.

Para ejemplificar esto utilizaremos el TSP, este problema consiste en encontrar un circuito de peso total mínimo, que atraviese todos los nodos de un grafo con pesos en las aristas (en este caso se busca un subconjunto de aristas). Asumiendo el peor caso donde el grafo es completo, si se empieza desde cualquier nodo, quedan $n - 1$ nodos para visitar, luego $n - 2$, luego $n - 3$, etc. Como en el caso general no importa el sentido en que se recorren las aristas, cada circuito factible vale por 2, puesto que se puede recorrer en 2 sentidos. Entonces la cantidad de soluciones factibles es $(n - 1)!/2$ donde n es la cantidad de nodos. Por lo tanto una instancia de TSP con $n = 20$ tendrá 60822550204416000 soluciones factibles. Si encontrar y evaluar cada una de estas soluciones demorara 0,001 segundos, evaluar todas demoraría aproximadamente 1928670 años.

Algunos problemas de optimización combinatoria bien conocidos son posibles de resolverse en forma exacta en tiempo polinomial, pero son un grupo muy reducido.

La comunidad científica se ha dedicado a lo largo del tiempo a atacar estos problemas a través de métodos más sofisticados como programación entera, programación lógica, algoritmos aproximados, heurísticas y metaheurísticas.

El problema que analizaremos, se puede plantear como de optimización combinatoria de la siguiente manera (no será la única):

- C es el conjunto de todos los partidos posibles entre todos los equipos en cada fecha.
- El conjunto de restricciones es tal que sólo lo satisfacen los subconjuntos de C que contienen (exactamente) todos los partidos de un fixture válido.
- f es una función que mide cuánto viajan en total los equipos en un fixture.

1.1.3. Programación lineal entera

Un problema de programación lineal (PPL) responde a la forma $\text{máx}\{cx : Ax \leq b, x \geq 0\}$, donde A es una matriz de $m \times n$, c es un vector-fila de dimensión n , y x es un vector columna de variables, también de dimensión n . La notación se lee “maximizar cx sujeto a $Ax \leq b$ ”. También puede presentarse como un problema de minimización.

Si los dominios de todas las variables son los números reales, estos problemas son resueltos muy eficientemente por el método simplex. Se ha demostrado que este algoritmo tiene una complejidad exponencial en peor caso, sin embargo en la práctica tiene un muy buen desempeño y es muy utilizado (inclusive aunque también exista un algoritmo polinomial en peor caso [12])

En cambio, si el dominio de las variables es discreto (0-1, naturales, enteros), estaremos ante la presencia de un problema de *programación lineal entera* (PPLE) [25]. Si se tienen una mezcla entre variables discretas y continuas, tendremos un *problema de programación lineal entera mixta* (PPLEM).

Estas sencillas restricciones sobre los dominios de las variables hacen que el problema sea NP-Completo, y no se lo pueda tratar directamente con el método simplex, puesto que este algoritmo sólo resuelve problemas con variables continuas.

Para solucionar este problema, se diseñaron algoritmos que utilizan simplex sobre relajaciones de los PPLE. Una *relajación* de un PPLE o PPLEM es el mismo problema, pero sin las condiciones de dominio discreto, con lo que se obtiene un PPL. Las relajaciones son fácilmente resueltas por simplex.

Son muy pocos los casos donde la solución de la relajación coincide con la del problema original. Sin embargo, una relajación donde haya suficientes variables con valores asignados en su dominio correspondiente, es más propensa a devolver un resultado válido para el problema original. En base a esto, distintos esquemas algorítmicos como son Branch & Bound, Branch & Cut, y Branch & Price [25], resuelven iterativamente relajaciones de los

problemas, fijando valores de variables en base a los resultados y volviendo a iterar hasta obtener soluciones válidas para el problema original.

1.1.4. Metaheurísticas y búsquedas locales

Cuando resolver un problema en forma exacta se vuelve demasiado complicado, una buena opción es usar *heurísticas*. Las heurísticas son algoritmos que resuelven el problema (llegan a soluciones factibles) pero que no garantizan optimalidad. Entra en esta categoría cualquier algoritmo que obtenga una solución factible para el problema. En la práctica, se han logrado obtener soluciones de muy buena calidad (ceranas al óptimo) para distintos problemas utilizando heurísticas. La ventaja que tienen las heurísticas por sobre los algoritmos exactos es que suelen requerir mucho menos tiempo y poder de procesamiento.

Las *metaheurísticas* son esquemas que agrupan diversas heurísticas, donde a partir de una solución inicial, se aplican criterios de refinamiento iterativamente, y si la solución mejora, se actualiza y se sigue iterando. Puesto que no se sabe si la mejor solución obtenida es óptima o no, se utilizan criterios de parada para finalizar el algoritmo, estos pueden ser límites para la cantidad de iteraciones totales o para la cantidad de iteraciones sin mejora. Algunas de las metaheurísticas más famosas son GRASP, Tabu Search, Simulated Annealing, Hill Climbing, Algoritmos Genéticos y Algoritmos de Colonias de Hormigas.

Para mejorar la solución en cada iteración, se le hacen pequeñas modificaciones que garanticen obtener otra solución factible (hay excepciones donde se exploran soluciones infactibles), luego se analiza si la nueva solución es mejor. Al criterio que define estas modificaciones se lo denomina *vecindario*, y a las soluciones obtenidas por éstas se las llama *vecinas* de la solución original. Una *búsqueda local* es el proceso de analizar el vecindario de una solución en busca de un vecino mejor.

Realizar iterativamente una búsqueda local no garantiza llegar al óptimo global del problema, de hecho la situación más probable es que se llegue a un *óptimo local*, una solución mejor que todas sus vecinas. Otra posibilidad es que la metaheurística se quede ciclando en un conjunto de soluciones igual de buenas. Para evitar quedarse en un óptimo local (o ciclar entre varios) se utilizan distintos recursos como usar distintos vecindarios en cada iteración o permitir elegir vecinos peores (a fin de explorar más el espacio de soluciones). Si se permite esto último, la metaheurística Tabu Search [11] propone mantener una *lista tabú*, la cual contendrá soluciones por las que ya se ha pasado anteriormente, con el fin de no repetirlas para evitar ciclar. Versiones más sofisticadas de Tabu Search prohíben aparte aplicar ciertos vecindarios o movimientos en base a mecanismos de aprendizaje utilizados en las iteraciones anteriores.

1.2. Características de campeonatos y fixtures

Dedicaremos esta sección a definir las distintas propiedades que pueden tener los campeonatos y fixtures en general. En este punto, distinguiremos los términos *campeonato* y *fixture*. Por campeonato nos referiremos a la liga deportiva en la que compiten los equipos, que consta de un conjunto fijo de equipos y se desarrolla a lo largo de un lapso de tiempo predeterminado. Llamaremos *fixture* a cada uno de los calendarios de partidos válidos para llevar a cabo el campeonato.

A lo largo de este trabajo, denominaremos n a la cantidad de equipos que participan en el campeonato, más adelante utilizaremos este número para medir la complejidad de los modelos y estimar la dificultad de resolverlos.

Tomaremos como convención que si se habla de un encuentro entre dos equipos, por ejemplo A y B, diremos que se juega “A vs B” y asumiremos que el primer equipo mencionado (en este caso A) es el local.

Uno de los esquemas de campeonato más comunes es el llamado *double round-robin*, en el cual el campeonato se divide en dos rondas. En cada una todos los equipos juegan contra todos una sola vez. Si en la primera ronda se juega A vs B, en la siguiente los roles se invertirán y se jugará B vs A. Un campeonato double round-robin consistirá de $2 \times (n - 1)$ fechas, puesto que cada equipo deberá jugar 2 veces contra todos los demás.

Un campeonato puede tener la propiedad de ser *espejado* (o *mirrored*), esto quiere decir que el fixture en la segunda mitad del campeonato es idéntico al de la primera, con la diferencia de que los roles de local y visitante están invertidos. Si un campeonato espejado tiene 18 fechas, y en la fecha 5 se juega A vs B, debe ocurrir que en la fecha 14 se juega B vs A.

Antes de que las ligas determinen el fixture, los equipos suelen expresar su deseo de jugar como local o visitante en alguna fecha por distintas razones. Estas condiciones serán conocidas como *restricciones de localía*.

En cada liga en particular, por razones de tiempo y económicas, los equipos pueden o no volver a su ciudad entre partidos de visitante seguidos. En el caso que analizaremos, los equipos no regresan, es por ello que nos interesa organizar el fixture lo mejor posible para disminuir las distancias que recorren. Recalcamos nuevamente que si el campeonato es double round-robin, y los equipos siempre vuelven a su ciudad después de un partido de visitante, no tiene sentido minimizar las distancias puesto que cada equipo visitará una vez (ida y vuelta) a todos los demás.

Es una propiedad deseable en los fixtures que los equipos no jueguen varios partidos seguidos de local o visitante, ya que los viajes podrían perjudicar a los equipos económicamente o por cansancio físico, inclusive se podría disminuir el atractivo del campeonato. A fin de evitar estas situaciones, se le suele prohibir al fixture que los equipos superen una cantidad determinada de partidos seguidos de local o visitante. Nos referiremos a este tipo de condi-

ciones como *restricciones de tramo*. Cuando los equipos vuelven a su ciudad después de cada partido de visitante, en vez de plantear las restricciones de tramo como inamovibles, se intenta minimizar la cantidad de estos tramos (*breaks*).

En el caso que estudiaremos, el campeonato se organiza *por parejas*. Antes de definir el fixture, se dividen los n equipos en $n/2$ parejas (el criterio usual es cercanía de ciudades). En este sistema los partidos se plantean en formato pareja vs pareja, y suelen denotar dos fechas en lugar de una, durante estas dos fechas, los equipos de la pareja local deberán enfrentarse a los de la visitante. Finalmente, en algún momento del campeonato se deberá dedicar una fecha para que los integrantes de las parejas se enfrenten entre sí. Para poder jugar un campeonato en parejas con formato round-robin sin modificaciones importantes (como dejar equipos libres en algunas fechas), la cantidad n de equipos debe ser divisible por 4. La razón de esto es muy sencilla, en cada fecha deberá haber $n/4$ partidos para que jueguen las $n/2$ parejas.

Una variación que analizaremos para los torneos por parejas es permitir que las parejas se recombinen entre weekends en lugar de mantenerse fijas durante todo el torneo. Si las parejas se sostienen durante todo el torneo, diremos que es un torneo con *parejas fijas*, si éstas pueden recombinarse, diremos que tiene *parejas dinámicas*.

En un campeonato por parejas, al enfrentarse dos de éstas deberán jugarse 4 partidos, si el orden en que se juegan esos partidos está predeterminado, diremos que el campeonato tiene un *esquema de visita fijo* (en la liga de voleibol el orden de los juegos está determinado por el ranking de los equipos en el campeonato anterior). Si los equipos de las parejas pueden enfrentarse en cualquier orden, hablaremos de un *esquema de visita dinámico*.

La secuencia de roles local/visitante por fecha que realiza un equipo en un fixture dado conforma su *patrón de localía* (o *home-away pattern* o HAP). Estos patrones son importantes ya que suelen ser un punto de partida a la hora de generar los fixtures.

1.3. El Travelling Tournament Problem (TTP)

El Travelling Tournament Problem fue formalmente propuesto por M. Trick, G. Nemhauser y K. Easton en el año 2001 ([9]).

Este problema modela un campeonato double round-robin de n equipos con restricciones de tramo (establece cotas superiores e inferiores para la cantidad de partidos seguidos en un mismo rol). El objetivo es obtener un fixture factible que minimice la suma de las distancias recorridas por los equipos. Se asume que los equipos deben comenzar y terminar el campeonato en sus respectivas ciudades. El problema recibe como entrada la matriz de distancias de $n \times n$ y las cotas superiores e inferiores de las restricciones de tramo. Usualmente se toma la cota inferior trivial 1 y se le da más importancia a la superior, que suele establecerse en 3.

Actualmente, el TTP es un problema de muy difícil resolución incluso para instancias pequeñas. Tal es la dificultad que M. Trick creó una página web [23] donde se desafía a la comunidad científica a resolver lo mejor posible distintos juegos de instancias del problema. Entre estas instancias, las que cobraron más relevancia fueron las correspondientes a la Liga Nacional de Hockey (NHL), las cuales son utilizadas a modo de benchmark en prácticamente todos los trabajos que tratan el problema.

Hasta la fecha, la instancia más grande que se ha logrado resolver óptimamente es de 10 equipos (NL10,[23]). En instancias más grandes, sólo se conocen soluciones factibles y cotas inferiores.

No hay una demostración formal de que el TTP sea NP-completo. Sin embargo, existen demostraciones de que determinadas variantes sí lo son. Schaerf A. demuestra en [21] que si se tiene un fixture donde no se especifican los equipos, el problema de asignar los equipos en los slots del fixture es NPC. Tampoco se ha encontrado una fórmula exacta para determinar el número de soluciones factibles para una instancia de TTP, pero es sabido que este número crece exponencialmente en función de la cantidad de equipos.

Los problemas de organización de fixtures, entre los que se incluye el TTP, han sido modelados, tratados y estudiados con una gran cantidad de técnicas de optimización combinatoria.

En [5],[6] y [7], de Werra modela con grafos el problema de encontrar fixtures minimizando breaks. Para ello representa los fixtures como factorizaciones de grafos completos en matchings. Cada nodo del grafo representa un equipo, y cada matching representará el conjunto de partidos de una fecha. A cada matching se lo hace dirigido para expresar los roles de local y visitante en los partidos. Esta idea también se aborda en [19].

Una práctica muy común es la de generar los fixtures incrementalmente [10, 21, 20]. Se generan primero patrones de localía (Home-Away Patterns, especificaciones que sólo dicen si los equipos son locales o visitantes en cada fecha). Estos patrones se usan como soluciones parciales y luego, en base a éstos, se resuelve el problema de completarlos hasta llegar a un fixture válido. En estos casos los primeros pasos para obtener soluciones parciales suelen hacerse con constraint programming, ya que este paradigma es más útil a la hora de obtener soluciones factibles, luego en las etapas finales se utiliza programación entera para finalizar la solución (excepto en [21], donde también se usa constraint programming).

En el campo de las metaheurísticas, se han implementado una gran variedad: algoritmos genéticos [14], colonias de hormigas [18], Clustering [15], GRASP [17], Simulated Annealing [22, 24], y Tabu Search[4, 8], entre otras. Las más exitosas han sido Simulated Annealing y Tabu Search. En la mayoría de las metaheurísticas se aborda la idea de que las “buenas soluciones” son relativamente similares. Por ello se intenta mejorar iterativamente una solución a través de búsquedas locales.

En lo que respecta al TTP, estos son los vecindarios en los que se trabaja:

- HAP prefijado [10, 21]: Un HAP determinado puede tener uno, muchos o ningún fixture que lo satisfaga. Dado un HAP sólo se requiere definir quién juega contra quién en cada fecha. Esta búsqueda local suele implementarse con programación entera.
- Intercambio de rivalidades entre equipos [17, 15, 24]: dados dos equipos A y B, que juegan en un campeonato con la propiedad de double round robin, seguro en una fecha se jugará “A local vs B visitante”, y en otra fecha “B local vs A visitante”. Estos dos partidos pueden intercambiarse entre sí para obtener otro fixture válido, lo que puede resultar en una menor distancia total recorrida. En el caso particular del campeonato espejado siempre se habla de intercambiar el partido con su correspondiente en la otra mitad. En las metaheurísticas de la literatura se prueba intercambiar de a dos equipos ya que intentar todas las variantes posibles (cantidad exponencial) no resulta factible.
- Intercambio de equipos [4, 15, 24]: consiste en intercambiar las posiciones de dos o más equipos dentro de un fixture dado. Es decir, si se intercambian los equipos A y B, A jugará todos los partidos de B y B jugará todos los partidos de A. En las metaheurísticas este proceso se suele hacer probando de a dos equipos.
- Intercambio de fechas [4, 15, 24]: se trata de intercambiar los conjuntos de partidos correspondientes a dos fechas del fixture, debe realizarse con un poco de cuidado en el caso que sea espejado puesto que a menos que se intercambien una fecha y su simétrica en la otra mitad, se estarán intercambiando cuatro fechas en total. De nuevo, a efectos de evitar la explosión combinatoria se suele probar intercambiar de a dos fechas.

En [16] y [4] se pueden encontrar recopilaciones más detalladas sobre el estado del arte del TTP.

1.4. Descripción del campeonato de voleibol

A continuación, describiremos en detalle el caso de estudio de este trabajo. El campeonato de la Liga Argentina de Voleibol Masculino tiene las siguientes propiedades:

1. Juegan 12 equipos divididos en 6 parejas fijas.
2. Se juega en 12 Weekends, separados en dos rondas de 6 donde cada pareja debe jugar contra todas las demás (*double round-robin*).
3. Si un equipo juega más de un weekend de visitante, no regresa a su ciudad entre weekends, viaja directamente a la ciudad del siguiente partido.

4. El fixture de la segunda ronda es igual a la primera pero con las localías invertidas (*campeonato espejado*).
5. Se asume que cada equipo empieza y termina cada mitad del campeonato en su ciudad.
6. El primer weekend de cada ronda es denominado *weekend interparejas* y está destinado a que los equipos de cada pareja jueguen una vez entre sí. El partido interpareja se dará en el Sábado de esa semana. Este weekend también respeta el campeonato espejado.
7. No se juegan más de dos weekends seguidos de local/visitante. Esta condición solo se aplica dentro de los 5 weekends de cada mitad que le siguen al interparejas.
8. Comportamiento de parejas en los demás weekends: si la pareja A-B juega contra C-D en un weekend, y A-B juega de local, entonces C y D visitarán el Jueves uno a A y el otro a B, luego el Sábado volverán a viajar (se cruzan, el que estaba en A viaja a B, y viceversa) para enfrentar al equipo local que les falta. El orden en que se juegan estos partidos se especifica en el siguiente ítem .
9. Los partidos entre dos parejas están ordenados por el ranking del campeonato anterior (esquema de visita fijo): si en un weekend se juega A-B vs C-D, entonces el Sábado se jugará el partido entre los dos de mayor ranking y el partido entre los dos de menor, o sea, $\max\text{Rank}(A,B)$ vs $\max\text{Rank}(C,D)$ y $\min\text{Rank}(A,B)$ vs $\min\text{Rank}(C,D)$, donde $\max\text{Rank}$ y $\min\text{Rank}$ denotan el equipo de mayor y menor ranking, respectivamente. El Jueves se juegan los otros dos partidos .

Se presentan (8) y (9) por separado ya que más adelante se hará mayor distinción entre estos conceptos.

En la Figura 1.1 mostramos a modo de ejemplo el caso de la liga 2007-2008. En este caso hay una pareja conformada por dos equipos de una misma ciudad (OSJ y UPC, ambos de San Juan).

1.5. Resumen de la tesis

En esta tesis abordamos la resolución del Travelling Tournament Problem con parejas de equipos con las condiciones adicionales de que (a) el esquema de visita es dinámico y (b) las parejas son dinámicas. De acuerdo con nuestro relevamiento de la literatura, estos problemas no han sido abordados hasta este momento, siendo la variante con esquema de visita estático y parejas fijas predeterminadas de antemano la única versión mencionada en trabajos anteriores [2].



Figura 1.1: Parejas de la Liga de Voleibol Primera División Masculina 2007-2008

En el Apéndice A, se pueden ver las definiciones de las instancias (equipos y ubicaciones de los mismos) que hemos usado para nuestros experimentos.

En el Capítulo 2 presentaremos modelos de programación entera cuya solución óptima contempla estas variantes. Estos modelos describen muy bien el problema, pero no son efectivos a la hora de obtener buenas soluciones en tiempo aceptable.

En el Capítulo 3 presentaremos un esquema algorítmico cuya finalidad es obtener soluciones aceptables. Para ello, utilizaremos versiones restringidas de los modelos del capítulo anterior, para resolver el problema en distintas etapas: primero definir una configuración de parejas fijas o dinámicas, luego obtener una solución con esquema de visita fijo, y, por último, mejorar la solución permitiendo esquema de visita dinámico. Para la última etapa de mejora, utilizaremos un esquema de metaheurística Tabu Search, con la innovación de que las búsquedas locales serán implementadas en base a modelos de programación entera.

En el Capítulo 4, se aprecian los resultados de estas técnicas, y en base a éstos se propondrán y analizarán ideas para mejorar el esquema algorítmico propuesto en el capítulo anterior.

Finalmente, en el Capítulo 5 presentaremos nuestras conclusiones y posible trabajo a futuro.

Capítulo 2

Modelos propuestos

En este capítulo, expondremos tres modelos de programación entera que representan el caso de la liga de voleibol con parejas dinámicas y un esquema de visita dinámico. Mostraremos también algunas variantes sencillas de los mismos.

2.1. Definiciones preliminares

A continuación haremos algunas definiciones que serán compartidas por todos los modelos:

- *Equipos* : conjunto de los n equipos que participan en el torneo, n debe ser múltiplo de 4.
- d_{ij} ($i, j \in Equipos$): distancia entre la ciudad del equipo i , y la del j .
- $id : Equipos \rightarrow \{1, \dots, n\}$: función que asigna un id único a cada equipo en un modelo. En principio se puede asumir que el id de un equipo será el ranking del torneo anterior.
- *Fechas* $\equiv \{1, \dots, 2n\}$: conjunto de las fechas que se juegan en el torneo. A diferencia del TTP donde se juegan $2(n - 1)$ fechas, se toman en cuenta las dos correspondientes a los Jueves de los weekends interparejas (se asume que cada equipo se queda en su ciudad).
- *Weekends* $\equiv \{1, \dots, n\}$: conjunto de los weekends que se juegan en el torneo.
- *Jueves* $\equiv \{1, 3, \dots, 2(n - 1)\}$: conjunto de las fechas Jueves del torneo, se corresponden con las fechas impares.
- *Sabados* $\equiv \{2, 4, \dots, 2n\}$: conjunto de las fechas Sábado del torneo, se corresponden con las fechas pares.

- $(Fechas/Weekends/Jueves/Sabados)1M$: conjunto que representa los elementos de $Fechas/Weekends/Jueves/Sabados$ que se corresponden con la primera mitad del torneo.
- $RWLocal \subseteq Equipos \times Weekends$: si $(e, w) \in RWLocal$ entonces el equipo e debe jugar de local en el weekend w .
- $RWVisitante \subseteq Equipos \times Weekends$: si $(e, w) \in RWVisitante$ entonces el equipo e debe jugar de visitante en el weekend w .
- $IJ1 \equiv 1$: Jueves del primer weekend interparejas.
- $IS1 \equiv 2$: Sábado del primer weekend interparejas.
- $IJ2 \equiv n + 1$: Jueves del segundo Weekend interparejas.
- $IS2 \equiv n + 2$: Sábado del segundo weekend interparejas.
- $IW1 \equiv 1$: primer weekend interparejas.
- $IW2 \equiv n/2 + 1$: segundo weekend interparejas.

2.2. Modelo 1

Este modelo es una extensión directa del modelo usual de TTP. Las variables P (y sus variantes) delimitan el problema para que el torneo sea por parejas.

Variables:

- X_{ijk} con $i, j \in Equipos, k \in (Fechas \cup \{2n + 1\})$
 $X_{ijk} = 1$ si en la fecha k , el equipo j juega de visitante contra el equipo i . Se utiliza X_{iik} para contar la vuelta a casa del equipo i . La fecha $2n + 1$ se utiliza para contabilizar el regreso de los equipos al finalizar el torneo.
- Y_{ijk} con $i, j \in Equipos, k \in Fechas$
 $Y_{ijk} = 1$ si entre la fecha k y la $k + 1$, algún equipo viaja de la ciudad de i a la de j . Notar que dada una fecha, a lo sumo un solo equipo puede recorrer ese tramo.
- P_{ijk} con $i, j \in Equipos, id(i) > id(j), k \in Jueves1M$
 $P_{ijk} = 1$ si el equipo i forma pareja con el equipo j , en la semana del jueves k .

Función objetivo:

$$\text{MIN} \sum_{\substack{i, j \in Equipos \\ f \in Fechas}} d_{ij} * Y_{ijf}$$

Restricciones:

1. Cada partido se juega una vez (en la primera mitad), se pide $id(i) > id(j)$ para evitar simetría:

$$\sum_{k \in Fechas1M} (X_{ijk} + X_{jik}) = 1$$

$$i, j \in Equipos, id(i) > id(j)$$

2. Cada equipo juega un partido por fecha (en la primera mitad):

$$\sum_{\substack{j \in Equipos \\ j \neq i}} (X_{ijk} + X_{jik}) = 1$$

$$i \in Equipos, k \in Fechas1M$$

3. Fijación de variables X_{iif} (1): caso de Jueves interpareja:

$$X_{iif} = 1$$

$$i \in Equipos, f \in \{IJ1, IJ2\}$$

4. Fijación de variables X_{iif} (2): Cada equipo vuelve a su ciudad al terminar el torneo:

$$X_{ii(2n+1)} = 1$$

$$i \in Equipos$$

5. Fijación de variables X_{iif} (3): en cualquier otro caso se anulan:

$$X_{iif} = 0$$

$$f \in Fechas - \{IJ1, IJ2, 2n + 1\}$$

6. Torneo espejado:

$$X_{ijk} = X_{ji(k+n)}$$

$$i, j \in Equipos, k \in Fechas1M$$

7. Un equipo no juega tres weekends seguidos de local (sin tomar en cuenta el primer weekend de cada mitad), se cuenta desde el segundo Jueves hasta el antepenúltimo (los dos restantes no pueden comenzar un tramo de tres seguidos) :

$$\sum_{\substack{j \in Equipos \\ j \neq i}} (X_{ijk} + X_{ij(k+2)} + X_{ij(k+4)}) \leq 2$$

$$i \in Equipos, k \in Jueves1M, IJ1 < f < n - 3$$

8. Un equipo no juega tres weekends seguidos de visitante (sin tomar en cuenta el primer weekend de cada mitad), se cuentan los Jueves de la misma forma :

$$\sum_{\substack{j \in \text{Equipos} \\ j \neq i}} (X_{jik} + X_{ji(k+2)} + X_{ji(k+4)}) \leq 2$$

$$i \in \text{Equipos}, k \in \text{Jueves}1M, IJ1 < f < n - 3$$

Restricciones de localía:

9. El equipo i debe jugar de local en el weekend w :

$$\sum_{\substack{j \in \text{Equipos} \\ j \neq i}} (X_{ji(2(w-1))} + X_{ji(2w)}) = 0$$

$$(i, w) \in RWLocal$$

10. El equipo i debe jugar de visitante en el weekend w :

$$\sum_{\substack{j \in \text{Equipos} \\ j \neq i}} (X_{ij(2(w-1))} + X_{ij(2w)}) = 0$$

$$(i, w) \in RWVisitante$$

Definición de Y_{ijk} :

11. El equipo q visita primero al equipo i , y luego al j :

$$Y_{ijk} \geq X_{iqk} + X_{jq(k+1)} - 1$$

$$i, j, q \in \text{Equipos}, i \neq j \neq q, k \in \text{Fechas}$$

12. El equipo i juega de local y luego visita al j :

$$Y_{ijk} \geq \sum_{\substack{q \in \text{Equipos} \\ q \neq j}} X_{iqk} + X_{ji(k+1)} - 1$$

$$i, j \in \text{Equipos}, i \neq j, k \in \text{Fechas}$$

13. El equipo j visita al i y luego juega de local:

$$Y_{ijk} \geq X_{ijk} + \sum_{\substack{q \in \text{Equipos} \\ q \neq i}} X_{jq(k+1)} - 1$$

$$i, j \in \text{Equipos}, i \neq j, k \in \text{Fechas}$$

14. Definición de los Y_{iik} , puesto que no aportan a la solución, se descartan estas variables:

$$Y_{iik} = 0$$

$$i \in Equipos, k \in Fechas$$

Comportamiento de parejas:

15. Equivalencia de las variables Y_{ijk} de Jueves a Sábado por la rotación de los visitantes:

$$Y_{ijk} = Y_{jik}$$

$$i, j \in Equipos, i \neq j, k \in Jueves, k > IJ1$$

16. Todo equipo que juega de local un Jueves, pertenece a una pareja local, por lo que deberá jugar de local el Sábado siguiente, no vale para el weekend interparejas:

$$\sum_{\substack{j \in Equipos \\ i \in Equipos, k \in Jueves1M, k > IJ1}} X_{ijk} = \sum_{j \in Equipos} X_{ij(k+1)}$$

17. Un equipo aparece en una pareja cada fecha:

$$\sum_{\substack{j \in Equipos \\ id(j) < id(i)}} P_{ijk} + \sum_{\substack{j \in Equipos \\ id(i) < id(j)}} P_{jik} = 1$$

$$i \in Equipos, k \in Jueves1M, k > IJ1$$

18. Si el equipo i juega de visitante contra el j un Jueves, y j y k son pareja, i juega con k el Sábado (dos restricciones):

$$P_{jkf} - 1 \leq X_{jif} + X_{ki(f+1)} \leq 1 - P_{jkf}$$

$$f \in Jueves1M, f > IJ1,$$

$$i, j, k \in Equipos, id(j) > id(k), i \neq j \neq k$$

19. Misma restricción, pero los partidos se juegan al revés (dos restricciones):

$$P_{jkf} - 1 \leq X_{kif} + X_{ji(f+1)} \leq 1 - P_{jkf}$$

$$f \in Jueves1M, f > IJ1,$$

$$i, j, k \in Equipos, id(j) > id(k), i \neq j \neq k$$

20. Si i juega de local contra j un Jueves, y j y k son pareja, i juega con k el Sábado (dos restricciones):

$$P_{jkf} - 1 \leq X_{ijf} + X_{ik(f+1)} \leq 1 - P_{jkf}$$

$$f \in Jueves1M, f > IJ1,$$

$$i, j, k \in Equipos, id(j) > id(k), i \neq j \neq k$$

21. Misma restricción, pero los partidos se juegan al revés (dos restricciones):

$$P_{j kf} - 1 \leq X_{j i f} + X_{k i (f+1)} \leq 1 - P_{j k f}$$

$$f \in \text{Jueves1M}, f > \text{IJ1},$$

$$i, j, k \in \text{Equipos}, id(j) > id(k), i \neq j \neq k$$

Variantes:

Existen una variedad de reformulaciones posibles para distintos aspectos de este modelo:

- Para que las parejas sean fijas durante todo el torneo se cambian los P_{ijk} por P_{ij} . Se replantean las siguientes restricciones:

- (17):
$$\sum_{\substack{j \in \text{Equipos} \\ id(j) < id(i)}} P_{ij} + \sum_{\substack{j \in \text{Equipos} \\ id(i) < id(j)}} P_{ji} = 1$$

 $i \in \text{Equipos}$

- (18):
$$P_{jk} - 1 \leq X_{j i f} - X_{k i (f+1)} \leq 1 - P_{jk}$$

 $f \in \text{Jueves1M}, f > \text{IJ1},$
 $i, j, k \in \text{Equipos}, id(j) > id(k), i \neq j \neq k$

- (19):
$$P_{jk} - 1 \leq X_{k i f} - X_{j i (f+1)} \leq 1 - P_{jk}$$

 $f \in \text{Jueves1M}, f > \text{IJ1},$
 $i, j, k \in \text{Equipos}, id(j) > id(k), i \neq j \neq k$

- (20):
$$P_{jk} - 1 \leq X_{i j f} - X_{i k (f+1)} \leq 1 - P_{jk}$$

 $f \in \text{Jueves1M}, f > \text{IJ1},$
 $i, j, k \in \text{Equipos}, id(j) > id(k), i \neq j \neq k$

- (21):
$$P_{jk} - 1 \leq X_{j i f} - X_{k i (f+1)} \leq 1 - P_{jk}$$

 $f \in \text{Jueves1M}, f > \text{IJ1},$
 $i, j, k \in \text{Equipos}, id(j) > id(k), i \neq j \neq k$

Por otro lado, también se eliminan las variables Y_{ijk} correspondientes a los tramos Jueves-Sábado (excepto IJ1 e IJ2), ya que esta información se puede obtener directamente de la configuración de los P_{ij} . Cada pareja jugará $n/2 - 1$ weekends de local en el torneo, durante los cuales las parejas visitantes recorrerán 2 veces la distancia entre los equipos locales. Por otro lado, en cada weekend interpareja, un equipo visitará a su pareja desde su ciudad, por lo que se cubre esa distancia dos veces más durante el torneo (de Jueves a Sábado). La función objetivo queda:

$$\text{MIN} \sum_{\substack{i, j \in \text{Equipos} \\ f \in \text{Sabados}}} d_{ij} * Y_{ijf} + \sum_{\substack{i, j \in \text{Equipos} \\ id(i) > id(j)}} P_{ij} * (n/2) * 2 * dist_{ij}$$

- Las variables P_{ijk} que indican si los equipos son pareja en ese weekend pueden ser reemplazadas por variables PL_{ijk} y PV_{ijk} . Estas variables expresan que los equipos formarán una pareja local o visitante en ese weekend. Básicamente:

$$P_{ijk} \leftrightarrow (PL_{ijk} \vee PV_{ijk})$$

Se replantean las siguientes restricciones:

- (17):
$$\sum_{\substack{j \in \text{Equipos} \\ id(j) < id(i)}} (PL_{ijk} + PV_{ijk}) + \sum_{\substack{j \in \text{Equipos} \\ id(i) < id(j)}} (PL_{jik} + PV_{ijk}) = 1$$

$$i \in \text{Equipos}, k \in \text{Jueves}, k > IJ1$$

- (18):

$$PL_{jkf} - 1 \leq X_{jif} - X_{ki(f+1)} \leq 1 - PL_{jkf}$$

$$f \in \text{Jueves}1M, f > IJ1,$$

$$i, j, k \in \text{Equipos}, id(j) > id(k), i \neq j \neq k$$

- (19):

$$PL_{jkf} - 1 \leq X_{kif} - X_{ji(f+1)} \leq 1 - PL_{jkf}$$

$$f \in \text{Jueves}1M, f > IJ1,$$

$$i, j, k \in \text{Equipos}, id(j) > id(k), i \neq j \neq k$$

- (20): $PV_{jkf} - 1 \leq X_{ijf} - X_{ik(f+1)} \leq 1 - PV_{jkf}$

$$f \in \text{Jueves}1M, f > IJ1,$$

$$i, j, k \in \text{Equipos}, id(j) > id(k), i \neq j \neq k$$

- (21):

$$PV_{jkf} - 1 \leq X_{jif} - X_{ki(f+1)} \leq 1 - PV_{jkf}$$

$$f \in \text{Jueves}1M, f > IJ1,$$

$$i, j, k \in \text{Equipos}, id(j) > id(k), i \neq j \neq k$$

- Es posible utilizar sólo PL o sólo PV ya que aún así se garantiza el comportamiento correcto de parejas. Exhibiremos el caso de PL , el de PV es análogo. Se replantea la siguiente restricción:

- (17) Esta vez un equipo puede estar en una pareja local en una fecha, o no:

$$\sum_{\substack{j \in \text{Equipos} \\ id(j) < id(i)}} PL_{ijk} + \sum_{\substack{j \in \text{Equipos} \\ id(i) < id(j)}} PL_{jik} \leq 1$$

$$i \in \text{Equipos}, k \in \text{Jueves}, k > IJ1$$

Se agrega la siguiente restricción:

22. Debe haber $n/4$ parejas locales por fecha:

$$\sum_{\substack{i,j \in \text{Equipos} \\ id(j) < id(i)}} PL_{ijk} = n/4$$

$$k \in \text{Jueves}, k > IJ1$$

Se eliminan las restricciones (20) y (21).

- Es posible especificar el comportamiento de las variables de pareja durante todo el torneo en vez de hacerlo sólo en la primera mitad (el torneo espejado hace correcta a la segunda automáticamente). Ello implica que en las restricciones (17), (18), (19), (20) y (21), f se cuantificará sobre *Jueves* en vez de *Jueves1M* (y aparte $f \notin \{IJ1, IJ2\}$).
- Si se usa PL_{ijk} (con o sin PV_{ijk}) también es posible descartar las variables Y_{ijk} correspondientes a los tramos Jueves-Sábado (excepto IJ1 e IJ2), y en cambio poner los PL_{ijk} en el objetivo de esta manera:

$$\text{MIN} \quad \sum_{\substack{i,j \in \text{Equipos} \\ f \in \text{Sabados} \cup \{IJ1, IJ2\}}} d_{ij} * Y_{ijf} + \sum_{\substack{i,j \in \text{Equipos} \\ id(i) > id(j) \\ k \in \text{Jueves} - \{IJ1, IJ2\}}} PL_{ijk} * 2 * dist_{ij}$$

Sin embargo, esto sólo funciona si se especifica a los PL en las dos mitades del torneo en las restricciones (18), (19), (20) y (21).

- Se puede plantear una versión alternativa de (18), (19), (20) y (21). En lugar de considerar la rotación de cada equipo por separado, se pide que el id del equipo que ha de rotar coincida en las fechas correspondientes. Nos referiremos a esta modificación del modelo como *versión ids*:

- (18):

$$(P_{jkf} - 1) * n \leq \sum_{i \in \text{Equipos}} (id(i) * X_{jif}) - \sum_{i \in \text{Equipos}} (id(i) * X_{ki(f+1)})$$

$$\leq (1 - P_{jkf}) * n$$

$$f \in \text{Jueves1M}, f > IJ1,$$

$$j, k \in \text{Equipos}, id(j) > id(k), j \neq k$$

- (19):

$$(P_{jkf} - 1) * n \leq \sum_{i \in \text{Equipos}} (id(i) * X_{kif}) - \sum_{i \in \text{Equipos}} (id(i) * X_{ji(f+1)})$$

$$\leq (1 - P_{jkf}) * n$$

$$f \in \text{Jueves1M}, f > IJ1,$$

$$j, k \in \text{Equipos}, id(j) > id(k), j \neq k$$

- (20):

$$\begin{aligned} & (P_{j_k f} - 1) * n \leq \\ & \sum_{i \in Equipos} (id(i) * X_{ijf}) - \sum_{i \in Equipos} (id(i) * X_{ik(f+1)}) \\ & \leq (1 - P_{j_k f}) * n \\ & f \in Jueves1M, f > IJ1, \\ & j, k \in Equipos, id(j) > id(k), j \neq k \end{aligned}$$
- (21):

$$\begin{aligned} & (P_{j_k f} - 1) * n \leq \\ & \sum_{i \in Equipos} (id(i) * X_{ikf}) - \sum_{i \in Equipos} (id(i) * X_{ij(f+1)}) \\ & \leq (1 - P_{j_k f}) * n \\ & f \in Jueves1M, f > IJ1, \\ & j, k \in Equipos, id(j) > id(k), j \neq k \end{aligned}$$

El número n se corresponde con la cantidad de equipos. Se asume que los ids van de 1 a n , puesto que sólo se usan para diferenciar los equipos entre sí y, en principio, se corresponden con un ranking de torneo anterior, no habrá problema en tomar esto por supuesto.

- Cualquier combinación de las anteriores. Hay que tomar en cuenta que si sólo se usa PL_{ijk} o PV_{ijk} no será posible dejar las parejas fijas durante todo el torneo puesto que habrá un caso (local o visitante) donde no se tendrá control sobre el equipo.

2.3. Modelo 2

Este modelo está basado en una representación del TTP sobre grafos distinta a la más revisada en la literatura (1-factorización de grafos completos [5, 6, 7, 19]). Esta representación es más similar al Travelling Salesman Problem.

Definiremos un digrafo pesado $G = (V, X)$ donde:

- $V = Equipos \times (Fechas \cup \{2n + 1\})$
- $X = \{(v_1, v_2) / fecha(v_1) < fecha(v_2)\}$, donde $fecha(v)$ denota la fecha correspondiente al nodo v .
- $peso(x) \in X = dist_{equipo(v_1), equipo(v_2)}$, donde $peso(x)$ representa el peso asignado a la arista x y $equipo(v)$ denota el equipo correspondiente al nodo v .

En este digrafo cada equipo i debe realizar un camino que cumpla como mínimo con las siguientes condiciones:

- Comienza en el nodo $(i, 1)$
- Termina en el nodo $(i, 2n + 1)$
- En las fechas donde corresponden partidos (es decir, excluyendo las fechas $IJ1, IJ2$ y $2n + 1$), debe haber un y sólo un equipo j que se encuentre con i en el mismo nodo.
- Si dos equipos se encuentran en un nodo v , éste debe corresponder a alguno de los dos equipos, denotando a $equipo(v)$ como el equipo local y al otro como visitante.
- En cada mitad del torneo, todos los equipos deben encontrarse una y sólo una vez.

Este es un grupo de restricciones mínimas para definir un fixture bajo los términos del digrafo. A continuación formularemos el modelo de programación entera en base a este digrafo, el cual representará la totalidad de las restricciones que componen el problema.

Variables:

- $Y_{ijkf} \quad i, j, k \in Equipos, f \in \{1 \dots 2n\}$
 $Y_{ijkf} = 1$ si entre la fecha f y la $f + 1$, el equipo k viaja de la ciudad de i a la de j . Notar que dada una fecha, a lo sumo un solo equipo puede recorrer ese tramo. Se agrega la fecha $2n$ para contabilizar el último viaje de cada equipo.

Función objetivo:

$$\text{MIN} \quad \sum_{\substack{i,j,k \in Equipos \\ f \in Fechas}} d_{ij} * Y_{ijkf}$$

Restricciones:

1. Los equipos parten de su ciudad al iniciar el torneo:

$$\sum_{j \in Equipos} Y_{kjk1} = 1$$

$$k \in Equipos$$

2. Los equipos vuelven a su ciudad al finalizar cada mitad:

$$\sum_{j \in Equipos} Y_{jkkq} = 1$$

$$k \in Equipos, q \in \{n, 2n\}$$

3. Un equipo k juega de local si y sólo si otro equipo distinto j va a visitarlo:

$$\sum_{q \in Equipos} Y_{qkkf} = \sum_{\substack{i,j \in Equipos \\ j \neq k}} Y_{ikjf}$$

$$k \in Equipos, f \in (Fechas - \{n, 2n\})$$

4. Si un equipo viaja a una ciudad en una fecha, debe salir de esa misma ciudad en la siguiente:

$$\sum_{k \in Equipos} Y_{kji} = \sum_{z \in Equipos} Y_{zj(f+1)}$$

$$i, j \in Equipos, f \in Fechas, f < 2n$$

5. Juegan todos contra todos, se pide a cada par de equipos que se encuentren una vez en la primera mitad:

$$\sum_{\substack{k \in Equipos \\ f \in Fechas \\ f < n}} Y_{kij} + \sum_{\substack{z \in Equipos \\ f \in Fechas \\ f < n}} Y_{zji} = 1$$

$$i, j \in Equipos, i \neq j$$

6. Un equipo no juega tres weekends seguidos de local:

$$\left(\sum_{j \in Equipos} Y_{jkk(f-1)} \right) + Y_{kkk(f+1)} + Y_{kkk(f+3)} \leq 2$$

$$k \in Equipos, f \in Jueves1M, IJ1 < f < n - 3$$

7. Un equipo no juega tres weekends seguidos de visitante:

$$\sum_{\substack{i, j \in Equipos \\ j \neq k}} (Y_{ijk(f-1)} + Y_{ijk(f+1)} + Y_{ijk(f+3)}) \leq 2$$

$$i \in Equipos, f \in Jueves1M, IJ1 < f < n - 3$$

8. Torneo espejado:

$$\sum_{k \in Equipos} Y_{kij} = \sum_{k \in Equipos} Y_{kji(f+n)}$$

$$i, j \in Equipos, i \neq j, f \in Fechas, f < IJ2$$

Restricciones de localía:

9. El equipo i debe jugar de local en el weekend w :

$$\sum_{\substack{j, k \in Equipos \\ k \neq i}} Y_{jki(2(w-1))} = 0$$

$$(i, w) \in RWLocal$$

10. El equipo i debe jugar de visitante en el weekend w :

$$\sum_{\substack{j \in Equipos \\ j \neq i}} Y_{jii(2(w-1))} = 0$$

$$(i, w) \in RWVisitante$$

Comportamiento de parejas:

11. Si un equipo juega de local un Jueves, juega de local el Sábado siguiente:

$$\sum_{j \in Equipos} Y_{jii(f-1)} = Y_{iii f}$$

$$i \in Equipos, f \in (Jueves - \{IJ1, IJ2\})$$

12. Si un equipo juega de visitante en un weekend, debe haber otro equipo con el que rote los rivales en el weekend:

$$Y_{jki f} \leq \sum_{q \in (Equipos - \{i, j, k\})} Y_{kjq f}$$

$$i, j, k \in Equipos, i \neq j \neq k, f \in (Jueves - \{IJ1, IJ2\})$$

Variantes:

Parejas fijas durante todo el torneo:

Se agregan las variables:

- P_{ij} $i, j \in \{1 \dots 12\}, i > j$
 $P_{ij} = 1$ si i y j son pareja durante todo el torneo.

La función objetivo puede ser expresada como antes, o de la siguiente manera:

$$\text{MIN} \sum_{\substack{i, j, k \in Equipos \\ f \in Sabados}} (d_{ij} * Y_{ijk f}) + \sum_{\substack{i, j \in Equipos \\ id(i) > id(j)}} (P_{ij} * (n/2) * 2 * dist_{ij})$$

Se agregan las restricciones:

13. Parejas activas por torneo:

$$\sum_{\substack{1 \leq i, j \leq 12 \\ i > j}} P_{ij} = n/2$$

14. Comportamiento de parejas locales de Jueves a Sábado (dos restricciones):

$$P_{ij} - 1 \leq Y_{ijq f} - Y_{jik f} \leq 1 - P_{ij}$$

$$i, j, q, k \in Equipos, i \neq j \neq q \neq k, i > j,$$

$$f \in Jueves1M, f \neq IJ1$$

15. Las parejas visitantes respetan la inversión de Jueves a Sábado (dos restricciones):

$$P_{ij} - 1 \leq Y_{qki f} - Y_{kqj f} \leq 1 - P_{ij}$$

$$i, j, q, k \in Equipos, i \neq j \neq q \neq k, i > j,$$

$$f \in Jueves1M, f \neq IJ1$$

2.4. Modelo 3

Este último es también en cierta manera una extensión del modelo de TTP común. Esta vez, en lugar de delimitar los fixtures por parejas en un modelo más general, los modelaremos directamente. Gracias a esto, este modelo será el más representativo del problema, aunque a costa de lidiar con un número de variables mucho mayor que los anteriores.

Variables:

- X_{ijqkw} con $i, j, q, k \in Equipos, id(i) > id(j),$
 $i \neq j \neq q \neq k, w \in (Weekends - \{IW1, IW2\})$
 $X_{ijqkw} = 1$ si en el weekend w, i y j juegan como pareja local contra q y k como pareja visitante. En ese weekend, el Jueves juegan i contra q y j contra k , el Sábado juegan i contra k y j contra q . Se pide $id(i) > id(j)$ porque ya se permiten todas las posibilidades de juego al permitir que se inviertan q y k . De aquí en adelante, se usará la notación

$$H(i, j, q, k, w) =$$

$$\{i, j, q, k \in Equipos, id(i) > id(j), i \neq j \neq q \neq k,$$

$$w \in (Weekends - \{IW1, IW2\})\}$$
- Z_{ijw} con $i, j \in Equipos, i \neq j, w \in \{IW1, IW2\}$
 $Z_{ijw} = 1$ si en el weekend interparejas w juega i de local contra j .
- Y_{ijw} con $i, j \in Equipos, w \in (Weekends - \{IW1, IW2\})$
 $Y_{ijw} = 1$ si del Sábado de w al Jueves de $w + 1$ algún equipo viaja de i a j .

Función Objetivo:

$$\text{MIN} \sum_{\substack{i,j \in \text{Equipos} \\ w \in \text{Weekends}}} (d_{ij} * Y_{ijw}) + \sum_{H(i,j,q,k,w)} (2 * d_{ij} * X_{ijqkw}) + \sum_{\substack{i,j \in \text{Equipos} \\ w \in \{IW1, IW2\}}} (d_{ij} * Z_{ijw})$$

Restricciones:

Las restricciones deben cubrir muchos casos de simetría por pedir $id(i) > id(j)$ en las X .

Notar que $H(i, j, q, k, w) \leftrightarrow H(i, j, k, q, w)$.

1. Cada partido se juega una vez en la primera mitad:

$$1 = \sum_{\substack{q,k \in \text{Equipos} \\ w \in (\text{Weekends1M} - \{IW1\}) \\ H(i,q,j,k,w)}} (X_{iqjkw} + X_{iqkjl}) + \sum_{\substack{q,k \in \text{Equipos} \\ w \in (\text{Weekends1M} - \{IW1\}) \\ H(q,i,j,k,w)}} (X_{qijkw} + X_{qikjl}) + \sum_{\substack{q,k \in \text{Equipos} \\ w \in (\text{Weekends1M} - \{IW1\}) \\ H(j,q,i,k,w)}} (X_{jqikw} + X_{jqkiw}) + \sum_{\substack{q,k \in \text{Equipos} \\ w \in (\text{Weekends1M} - \{IW1\}) \\ H(q,j,i,k,w)}} (X_{qjikw} + X_{qjkiw}) + (Z_{ijIW1} + Z_{jiIW1})$$

$i, j \in \text{Equipos}, i \neq j$

2. Cada equipo juega un partido por weekend (en realidad dos):

$$1 = \sum_{\substack{j,q,k \in \text{Equipos} \\ H(i,j,q,k,w)}} X_{ijqkw} + \sum_{\substack{j,q,k \in \text{Equipos} \\ H(j,i,q,k,w)}} X_{jiqkw} + \sum_{\substack{j,q,k \in \text{Equipos} \\ H(j,q,i,k,w)}} (X_{jqikw} + X_{jqkiw})$$

$i \in \text{Equipos}, w \in (\text{Weekends1M} - \{IW1\})$

3. Cada equipo juega un partido en el primer weekend interpareja (ajuste de la restricción anterior para las Z):

$$\sum_{\substack{j \in \text{Equipos} \\ j \neq i}} (Z_{ijIW1} + Z_{jiIW1}) = 1$$

$i \in \text{Equipos}$

4. Torneo espejado para la segunda mitad (dos casos):

$$\begin{aligned}
X_{ijqkw} &= X_{qkij(w+n/2)} \\
i, j, q, k &\in \text{Equipos}, w \in (\text{Weekends1M} - \{IW1\}), \\
H(i, j, q, k, w), id(q) &> id(k) \\
X_{ijqkw} &= X_{kqji(w+n/2)} \\
i, j, q, k &\in \text{Equipos}, w \in (\text{Weekends1M} - \{IW1\}), \\
H(i, j, q, k, w), id(q) &< id(k)
\end{aligned}$$

5. Ajuste de la restricción anterior para las Z :

$$\begin{aligned}
Z_{ijIW1} &= Z_{jiIW2} \\
i, j &\in \text{Equipos}, i \neq j
\end{aligned}$$

6. Un equipo no juega tres weekends seguidos de local (sin tomar en cuenta el primer weekend de cada mitad), se cuenta desde el segundo Weekend hasta el antepenúltimo (los dos restantes no pueden comenzar un tramo de tres seguidos) :

$$\begin{aligned}
&\sum_{\substack{j,q,k \in \text{Equipos} \\ H(i,j,q,k,w)}} X_{ijqkw} + \sum_{\substack{j,q,k \in \text{Equipos} \\ H(j,i,q,k,w)}} X_{jiqkw} + \\
&\sum_{\substack{j,q,k \in \text{Equipos} \\ H(i,j,q,k,w)}} X_{ijqk(w+1)} + \sum_{\substack{j,q,k \in \text{Equipos} \\ H(j,i,q,k,w)}} X_{jiqk(w+1)} + \\
&\sum_{\substack{j,q,k \in \text{Equipos} \\ H(i,j,q,k,w)}} X_{ijqk(w+2)} + \sum_{\substack{j,q,k \in \text{Equipos} \\ H(j,i,q,k,w)}} X_{jiqk(w+2)} \\
&\leq 2 \\
&i \in \text{Equipos}, w \in \text{Weekends1M}, 1 < w < n/2 - 1
\end{aligned}$$

7. Un equipo no juega tres weekends seguidos de visitante (sin tomar en cuenta el primer weekend de cada mitad), se cuentan los Weekends de la misma forma :

$$\begin{aligned}
&\sum_{\substack{j,q,k \in \text{Equipos} \\ H(q,k,i,j,w)}} (X_{qkijw} + X_{qkjiw}) + \\
&\sum_{\substack{j,q,k \in \text{Equipos} \\ H(q,k,i,j,w)}} (X_{qkij(w+1)} + X_{qkji(w+1)}) + \\
&\sum_{\substack{j,q,k \in \text{Equipos} \\ H(q,k,i,j,w)}} (X_{qkij(w+2)} + X_{qkji(w+2)}) \\
&\leq 2 \\
&i \in \text{Equipos}, w \in \text{Weekends1M}, 1 < w < n/2 - 1
\end{aligned}$$

Restricciones de localía:

8. El equipo
- i
- debe jugar de local en el weekend
- w
- :

$$\sum_{j,q,k \in Equipos} (X_{jqkiw} + X_{jqikw}) = 0$$

$$(i, w) \in RWLocal$$

9. El equipo
- i
- debe jugar de visitante en el weekend
- w
- :

$$\sum_{\substack{j,q,k \in Equipos \\ H(i,j,q,k,w)}} X_{ijqkw} + \sum_{\substack{j,q,k \in Equipos \\ H(j,i,q,k,w)}} X_{jiqkw} = 0$$

$$(i, w) \in RWVisitante$$

Definición de las Y :

10. El equipo
- i
- juega de visitante contra
- j
- en el Sábado de
- w
- y en el weekend siguiente, de visitante contra
- q
- (el Jueves):

$$Y_{jqw} \geq \sum_{\substack{k,t \in Equipos \\ H(j,k,i,t,w)}} X_{jktiw} + \sum_{\substack{k,t \in Equipos \\ H(k,j,i,t,w)}} X_{kjitw} + \sum_{\substack{k,t \in Equipos \\ H(q,k,i,t,w)}} X_{qkit(w+1)} + \sum_{\substack{k,t \in Equipos \\ H(k,q,i,t,w)}} X_{kqit(w+1)} - 1$$

$$i, j, q \in Equipos, i \neq j \neq q, w \in (Weekends - \{IW1, n/2, IW2\})$$

11. Ajuste de la restricción anterior para las
- Z
- :

$$Y_{jqw} \geq Z_{jif} + \sum_{\substack{k,t \in Equipos \\ H(q,k,i,t,(w+1))}} X_{qkit(w+1)} + \sum_{\substack{k,t \in Equipos \\ H(q,k,t,i,(w+1))}} X_{qkti(w+1)} - 1$$

$$i, j, q \in Equipos, i \neq j \neq q, w \in \{IW1, IW2\}$$

12. El equipo
- i
- juega de local el Sábado, y luego
- i
- juega de visitante contra

$$j:$$

$$Y_{ijw} \geq \sum_{\substack{q,k,t \in Equipos \\ H(i,k,q,t,w)}} X_{iktqw} + \sum_{\substack{q,k,t \in Equipos \\ H(k,i,q,t,w)}} X_{kitqw} + \sum_{\substack{k,t \in Equipos \\ H(j,k,i,t,w)}} X_{jkit(w+1)} + \sum_{\substack{k,t \in Equipos \\ H(k,j,i,t,w)}} X_{kjit(w+1)} - 1$$

$$i, j \in Equipos, i \neq j, w \in (Weekends - \{IW1, n/2, IW2\})$$

13. Ajuste de la restricción anterior para las Z :

$$Y_{ijw} \geq \sum_{\substack{q \in \text{Equipos} \\ q \neq i}} Z_{iqw} + \sum_{\substack{q, k \in \text{Equipos} \\ H(j, k, i, t, w)}} X_{jkqi(w+1)} + \sum_{\substack{q, k \in \text{Equipos} \\ H(k, j, i, t, w)}} X_{kjqi(w+1)} - 1$$

$i, j \in \text{Equipos}, i \neq j, w \in \{IW1, IW2\}$

14. El equipo i juega de visitante contra j el Sábado, y luego i juega de local contra algún q :

$$Y_{jif} \geq \sum_{\substack{k, t \in \text{Equipos} \\ H(j, k, t, i, w)}} X_{jktiw} + \sum_{\substack{k, t \in \text{Equipos} \\ H(k, j, i, t, w)}} X_{kjitw} + \sum_{\substack{q, k, t \in \text{Equipos} \\ H(i, k, q, t, w)}} X_{ikqt(w+1)} + \sum_{\substack{q, k, t \in \text{Equipos} \\ H(k, i, t, q, w)}} X_{kitq(w+1)} - 1$$

$i, j \in \text{Equipos}, i \neq j, w \in (\text{Weekends} - \{IW1, n/2, IW2\})$

15. Ajuste de la restricción anterior para las Z :

$$Y_{jif} \geq Z_{jiw} + \sum_{\substack{q, k, t \in \text{Equipos} \\ H(i, k, q, t, w)}} X_{ikqt(w+1)} + \sum_{\substack{q, k, t \in \text{Equipos} \\ H(k, i, t, q, w)}} X_{kitq(w+1)} - 1$$

$i, j \in \text{Equipos}, i \neq j, w \in \{IW1, IW2\}$

16. Definición de los Y_{iik} , puesto que no aportan a la solución, se descartan estas variables:

$$Y_{iiv} = 0$$

$i \in \text{Equipos}, k \in \text{Weekends}$

17. Regreso a casa en cada mitad:

$$Y_{ijw} = \sum_{\substack{q, k \in \text{Equipos} \\ H(i, q, k, j, w)}} X_{iqkjw} + \sum_{\substack{q, k \in \text{Equipos} \\ H(q, i, k, j, w)}} X_{qikjw}$$

$i, j \in \text{Equipos}, i \neq j, w \in \{n/2, n\}$

Variantes:

Parejas fijas durante todo el torneo:

Se agregan las variables:

- P_{ij} $i, j \in \{1 \dots 12\}, i > j$
 $P_{ij} = 1$ si i y j son pareja durante todo el torneo.

La función objetivo puede ser expresada como antes, o de la siguiente manera:

$$\text{MIN} \sum_{\substack{i,j \in \text{Equipos} \\ w \in \text{Weekends}}} (d_{ij} * Y_{ijw}) + \sum_{\substack{i,j \in \text{Equipos} \\ id(i) > id(j)}} P_{ij} * (n/2) * 2 * dist_{ij}$$

Se agregan las restricciones:

18. Parejas activas por torneo:

$$\sum_{\substack{i,j \in \text{Equipos} \\ id(i) > id(j)}} P_{ij} = 6$$

19. Efecto de los P sobre las X , si i y j son pareja en el torneo, deberán jugar 10 partidos contra otras parejas:

$$P_{ij} * 10 = \sum_{\substack{q,k \in \text{Equipos} \\ w \in \text{Weekends} \\ H(i,j,q,k,w)}} X_{ijqkw} + \sum_{\substack{q,k \in \text{Equipos} \\ w \in \text{Weekends} \\ H(q,k,i,j,w)}} (X_{qkijw} + X_{qkjiw})$$

$i, j \in \text{Equipos}, id(i) > id(j)$

2.5. Comparación de los modelos

Más allá del hecho de que analizamos situaciones donde n está fijo en 12, tomaremos n para comparar los modelos, contemplando el caso de que se agreguen más equipos en la liga.

2.5.1. Variables

En cada modelo se pueden eliminar distintos subconjuntos de variables sin afectar significativamente los resultados expuestos a continuación.

Recordar que $\#Equipos = n$, $\#Fechas = 2n$, $\#Jueves1M = \frac{n}{2}$ y $\#Weekends = n$.

La cantidad de variables del Modelo 1 es:

$$\#Vars1 = \#X_{ijk} + \#Y_{ijk} + \#P_{ijk}$$

$$\text{donde } \#X_{ijk} = \#Equipos \times \#Equipos \times (\#Fechas + 1) = 2n^3 + n^2$$

$$\#Y_{ijk} = \#Equipos \times \#Equipos \times \#Fechas = 2n^3$$

$$\#P_{ijk} = \frac{\#Equipos \times (\#Equipos - 1)}{2} \times \#Jueves1M = \frac{n^3 - n^2}{4}$$

Luego, $\#Vars1 = \frac{17}{4}n^3 + \frac{3}{4}n^2$. Por lo tanto podemos decir que la cantidad de variables del Modelo 1 es $O(n^3)$. En el caso de que se usen variables PL y PV el resultado es similar, pero con el doble de variables de pareja.

La cantidad de variables del Modelo 2 es:

$$\#Vars2 = \#Y_{ijkf}$$

donde

$n/modelo$	1	2	3
4	284	512	80
8	2224	8192	5536
12	7452	41472	61104
16	17600	131072	309824

Tabla 2.1: Comparación de cantidad de variables en función de la cantidad de equipos.

$$\#Y_{ijkf} = \#Equipos \times \#Equipos \times \#Equipos \times (\#Fechas) = 2n^4$$

Entonces la cantidad de variables del Modelo 2 es $O(n^4)$.

La cantidad de variables del Modelo 3 es:

$$\#Vars3 = \#X_{ijkf} + \#Y_{ijk} + \#Z_{ijf}$$

donde

$$\begin{aligned} \#X_{ijk} &= \\ \frac{\#Equipos \times (\#Equipos - 1)}{2} \times (\#Equipos - 2) \times (\#Equipos - 3) \times (\#Weekends - 2) &= \\ = \frac{n^2 - n}{2} \times (n - 2) \times (n - 4) \times (n - 2) &= \\ = \frac{1}{2}n^5 - 4n^4 + \frac{23}{2}n^3 - 14n^2 + 6n & \end{aligned}$$

$$\#Z_{ijk} = \#Equipos \times (\#Equipos - 1) \times \#\{IW1, IW2\} = 2n^2 - 2n$$

$$\#Y_{ijk} = \#Equipos \times \#Equipos \times (\#Weekends - 2) = n^3 - 2n^2$$

Finalmente, $\#Vars3 = \frac{1}{2}n^5 - 4n^4 + \frac{25}{2}n^3 - 14n^2 + 4n$. En cantidad de variables este es el modelo más pesado puesto que es $O(n^5)$. Para el caso real que estudiaremos, con $n = 12$, la cantidad de variables X_{ijk} llega a las 59400. En la Tabla 2.1 se puede ver una comparativa de las cantidades de variables en función de la cantidad de equipos

2.5.2. Restricciones

Ahora analizaremos el número de restricciones con que cuentan los modelos. Tomando en cuenta que estamos haciendo un análisis de magnitudes en peor caso, nos enfocaremos en las familias de restricciones más grandes.

En el Modelo 1, hay 5 familias de restricciones que son $O(n^4)$, estas son:

- La restricción (3.1) que activa los Y_{ijk} en el caso donde un equipo es visitante dos veces seguidas .
- Las restricciones (18), (19), (20), (21) de comportamiento de pareja .

En el caso de las últimas, es posible reducir su número a $O(n^3)$ si se reemplazan por las versiones en base a los ids de los equipos (ver modificaciones

Modelo	#Variables	#Restricciones
1	$O(n^3)$	$O(n^4)$
2	$O(n^4)$	$O(n^4)$
3	$O(n^5)$	$O(n^4)$

Tabla 2.2: Comparación de cotas para variables y restricciones de los modelos.

del modelo). Esta característica es lo que hace más atractiva a esa versión del modelo. De todas maneras, en base a la familia (3.1), podemos afirmar que la cantidad de restricciones de este modelo es $O(n^4)$.

En lo que al Modelo 2 original respecta, la familia que modela el comportamiento de parejas (12) es la más numerosa de todas y hace que el número de restricciones sea $O(n^4)$. Sin embargo, si se emplea la extensión para parejas fijas, la cantidad de restricciones pasa a ser $O(n^5)$, a causa de las familias (14) y (15).

Finalmente, las restricciones del Modelo 3 son $O(n^4)$, esta vez por las restricciones (4) de fixture espejado y por la familia (10) (análoga a la (3.1) del Modelo 1).

En la Tabla 2.2 resumimos los resultados obtenidos hasta el momento.

2.5.3. Efectividad de los modelos

Lamentablemente ninguno de los tres modelos logra llegar al óptimo (siquiera a una solución factible) en un tiempo aceptable. Usualmente no superan un gap de 70 % en corridas de más de un día entero. Claramente esta situación no es buena, por lo que propondremos alternativas para obtener soluciones en base a estos mismos modelos, restringiendo y relajando distintos aspectos del problema.

Capítulo 3

Estrategias de resolución

Como vimos en el capítulo anterior, los tiempos de resolución para los modelos planteados hacen que resulte imposible resolver el problema directamente por medio de paquetes de programación lineal entera. Será necesario entonces aplicar distintas estrategias a fin de conseguir buenas soluciones en un tiempo aceptable.

El TTP plano ya de por sí es un problema de difícil resolución, hasta la fecha no se ha logrado resolver de forma exacta instancias de 12 equipos. Las condiciones extra que exigen los modelos para asegurar el formato de parejas, lejos de acelerar la resolución del problema, lo hacen más difícil.

Si se definen 6 parejas fijas para el torneo y se usa un esquema de visita fijo, lo que se obtiene es un TTP (con una ligera variación) de $n/2$ parejas en vez de equipos. De esta forma se obtiene un problema muy simplificado (un TTP con $n = 6$ equipos) capaz de resolverse óptimamente en poco tiempo. Este es el caso original del torneo y es la manera en que se estuvieron generando sus fixtures en los últimos años. La desventaja que se da en este caso es que a efectos de relajar el esquema de visita o el esquema de parejas, esa simplificación deja de ser válida.

En este capítulo estudiaremos distintas estrategias de resolución del problema. Comenzaremos por repasar el modelo que se estuvo usando para crear los fixtures en los últimos años. Luego proseguiremos con nuestra propuesta, donde dividiremos el problema en subproblemas que hagan énfasis en distintos aspectos del principal, para luego usarlos en conjunto para obtener una solución de calidad en un tiempo aceptable.

3.1. Procedimiento original

Para obtener un fixture según el sistema actual de la liga, se puede reducir el problema a una variante del TTP de $n/2$ equipos de la siguiente manera:

- Se utiliza el Modelo 1, sin variables P .
- Las variables Y_{ijw} se duplican, ya que deben cubrir dos maneras de

realizar los viajes.

- Cada pareja (definida previamente) cuenta como un equipo.
- Los weekends cuentan como fechas.
- Se asume que los partidos interparejas se juegan en un orden determinado (por ejemplo, el primer weekend interparejas juega de local el equipo de mayor ranking).
- Si i es una pareja conformada por los equipos A y B , definiremos d_i como d_{AB}
- La distancia entre dos parejas entre un weekend interparejas y el siguiente normal para la pareja visitante queda definida por :

$$dV_{(AB)(CD)} = d_{QC} + d_{QD}$$
donde Q (A o B) es el equipo que juega de local en el weekend interparejas correspondiente. En base a este cálculo se arman dos matrices de distancias en base a los equipos que juegan de local en cada weekend interpareja. Estos valores serán los coeficientes en el objetivo de las variables YV_{ijw} con $w \in \{IW1, IW2\}$. En este caso, los $YL_{(AB)(AB)w}$ ($w \in \{IW1, IW2\}$) sí estarán activos y su coeficiente será d_{AB} .
- La distancia entre dos parejas en los weekends intermedios, cuando una pareja viaja a visitar a otra, queda definida por el esquema de visita:

$$dV_{(AB)(CD)} = d_{MaxRank(A,B), MaxRank(C,D)} + d_{MinRank(A,B), MinRank(C,D)}$$
Se arma una matriz en base a estas distancias. Estos valores serán los coeficientes en el objetivo de las variables YV_{ijw} con $w \notin \{IW1, IW2\}$.
- La distancia entre dos parejas en los weekends intermedios cuando una pareja vuelve a su hogar queda definida a la inversa, sea AB la pareja que regresa:

$$dL_{(CD)(AB)} = d_{MinRank(A,B), MaxRank(C,D)} + d_{MaxRank(A,B), MinRank(C,D)}$$
Se arma una matriz en base a estas distancias. Estos valores serán los coeficientes en el objetivo de las variables YL_{ijw} con $w \notin \{IW1, IW2\}$.
- Cada $X_{(AB)(CD)k}$ tiene como coeficiente en la función objetivo $2d_{AB}$, excepto las $X_{(AB)(AB)f}$ de los weekends interpareja que tienen como coeficiente: d_{AB} .
- En los weekends interpareja se activan $X_{(AB)(AB)f}$ con peso d_{AB} .

Variables:

- X_{ijk} con $i, j \in Parejas, k \in Weekends$
 $X_{ijk} = 1$ si en la fecha k , la pareja j juega de visitante contra la pareja i en el Weekend k .

- YV_{ijk} con $i, j \in Parejas, i \neq j, k \in Fechas$
 $YV_{ijk} = 1$ si entre el weekend k y el $k + 1$, alguna pareja distinta de j viaja de las ciudades de i a la de j .
- YL_{ijk} con $i, j \in Parejas, i \neq j, k \in Fechas$
 $YL_{ijk} = 1$ si entre el weekend k y el $k + 1$, la pareja j viaja de las ciudades de i a las suyas.

Restricciones:

Las restricciones que no incluyen a las variables Y se toman tal cual aparecen en el Modelo 1.

- La pareja q visita primero a la pareja i , y luego a la j :

$$YV_{ijk} \geq X_{iqk} + X_{jq(k+1)} - 1$$

$$i, j, q \in Parejas, i \neq j \neq q, k \in Weekends$$

- El equipo i juega de local y luego visita al j :

$$YV_{ijk} \geq \sum_{\substack{q \in Equipos \\ q \neq j}} X_{iqk} + X_{ji(k+1)} - 1$$

$$i, j \in Parejas, i \neq j, k \in Weekends$$

- El equipo j visita al i y luego juega de local:

$$YL_{ijk} \geq X_{ijk} + \sum_{\substack{q \in Equipos \\ q \neq i}} X_{jq(k+1)} - 1$$

$$i, j \in Parejas, i \neq j, k \in Weekends$$

Función objetivo:

$$\begin{aligned} \text{MIN} \quad & \sum_{\substack{i, j \in Parejas \\ i \neq j \\ f \in Weekends}} (dV_{ij} * YV_{ijf} + dL_{ij} * YL_{ijf}) + \\ & \sum_{\substack{i, j \in Parejas \\ f \in Weekends - \{IW1, IW2\}}} (2 * d_i * X_{ijf}) + \sum_{\substack{i \in Parejas \\ f \in \{IW1, IW2\}}} (d_i * X_{iif}) \end{aligned}$$

En este modelo, de forma idéntica al Modelo 3, se les asigna un peso a las variables X_{ijk} , representativo de las distancias que recorren los visitantes entre Jueves y Sábado. A las X_{ijk} correspondientes a los weekends interpareja solo se les asigna la mitad de esa distancia puesto que en ese caso un sólo equipo hace el viaje.

Por otro lado, se deben calcular distintos juegos de distancias entre parejas, dos para los casos en que se sale de cada weekend interparejas, y dos

para el caso general de dos weekends cualesquiera en el resto del torneo. Notar también que si una pareja juega de local luego de un weekend interparejas, debe contabilizarse que el equipo visitante en la interparejas debe regresar a su estadio, para ello se utilizan las variables Y_{iww} con $w \in \{IW1, IW2\}$.

El modelo resultante puede resolverse en pocos minutos. Sin embargo, no es posible aplicar el esquema de visita dinámico a este modelo, ya que asume que las posiciones de los equipos al iniciar y terminar un weekend están determinadas, tampoco permite que se dinamice el esquema de parejas, puesto que las considera como unidades-equipos. La adaptación más directa para poder contemplar todas estas cuestiones es lo que ya hemos presentado como Modelo 3.

3.2. Nuestra propuesta

En las siguientes secciones describiremos distintos modelos, los cuales nos permitirán obtener gradualmente una buena solución, contemplando primero (y opcionalmente) la incorporación de parejas dinámicas y luego el esquema de visita dinámico. El procedimiento que usaremos será el siguiente:

1. Definir una *configuración de parejas* para el fixture, o sea, una especificación de qué equipos conforman pareja en cada weekend del torneo. Esta configuración puede ser de parejas fijas o dinámicas, y fijará este aspecto del problema en el procedimiento.
2. En base a esa configuración, obtener un buen fixture con esquema de visita fijo.
3. Mejorar iterativamente el fixture anterior relajando el esquema de visita.

Analizaremos este procedimiento para generar fixtures con parejas fijas y fixtures con parejas dinámicas.

3.2.1. Definición de la configuración de parejas

La propiedad de torneo por parejas es la más característica y condicionante del problema que analizamos. Como se mencionó en el capítulo introductorio, actualmente las parejas se establecen de forma fija durante el torneo, sin herramientas formales, bajo un criterio de cercanía general entre los equipos que conforman la pareja. Sin embargo, pueden darse diversas situaciones por las que no resulte factible aparear dos equipos A y B:

- Existe una fecha donde el equipo A debe jugar de visitante y el B de local, o viceversa.

- Si A debe jugar los weekends w y $w + 1$ de local, y B debe jugar de visitante $w + 3$ $w + 4$, resulta imposible que A y B sean pareja sin violar las restricciones de tramo.
- Se establece arbitrariamente que A y B no pueden ser pareja por alguna razón (por ejemplo para evitar hacer un recorrido peligroso en términos de seguridad).

En base a esto, desarrollaremos un modelo que nos provea de una configuración de parejas automáticamente. Llamaremos a este modelo *Modelo de Configuración de Parejas* (MCP de aquí en adelante).

Si se toma el Modelo 3 y se le quita las variables Y_{ijk} , lo que se obtiene es un modelo que resolverá el problema pero sólo tomando en cuenta las distancias recorridas entre Jueves y Sábado por las parejas visitantes. La principal ventaja que se obtiene es que este modelo llega al óptimo en pocos minutos.

El resultado devuelto puede ser muy malo si se lo toma como una solución completa, ya que se ignoran totalmente los viajes Sábado-Jueves, sin embargo, en la práctica da buenos resultados a efectos de definir una configuración de parejas (estática o dinámica) para el torneo. Por otro lado no es incorrecto pensar en minimizar primero las distancias que han de ser recorridas de Jueves a Sábado que las de Sábado a Jueves, puesto que se dispone de un tiempo mucho más corto para recorrerlas. La configuración obtenida será utilizada por otros modelos para generar una solución de calidad.

Cabe mencionar que a efectos de este problema, el esquema de visita no ha de ser tomado en cuenta, puesto que lo que realmente se evalúa es la distancia entre las parejas locales.

En el caso de un torneo donde no haya restricciones de localía, resolver este problema equivale a buscar un matching mínimo en un grafo completo donde haya un nodo por equipo y las aristas tengan por peso el doble de las distancias entre los nodos. Ante la presencia de restricciones de localía, esto podría no ser cierto, éstas, junto con las restricciones de tramo, podrían prohibir el caso de formar dos parejas (entre 4 equipos) cuyos patrones de localía fueran incompatibles para enfrentarse entre sí.

Es posible expresar este modelo a partir del Modelo 1, eliminando solamente las variables Y_{ijk} con $k \in \text{Sabados}$, o, preferiblemente, utilizar alguna variante con variables PL_{ijk} y eliminar la totalidad de las Y_{ijk} (que serán solamente las que cumplan $k \in \text{Sabados}$). En el caso de parejas fijas es posible expresar la función objetivo sólo en función de los P_{ij} .

A diferencia de los otros dos, el Modelo 2 no es idóneo para simplificarse de esta manera. La razón es que no es posible eliminar las variables de viaje innecesarias más allá de quitarlas del objetivo puesto que éstas son parte estructural de la solución.

Indistintamente de si se parte del Modelo 1 o 3, ambas posibilidades permiten obtener un esquema de parejas dinámico o estático (utilizando la variante de parejas fijas correspondientes).

Dependiendo del modelo que se resuelva, se obtendrá una de estas dos soluciones:

- Configuración de parejas fija: $\frac{n}{2}$ parejas predefinidas para todo el torneo
- Configuración de parejas dinámica: especificación de $\frac{n}{2}$ parejas por cada weekend.

En ambos casos, para cualquier solución del problema completo que respete la configuración, el peso total de los viajes Jueves-Sábado se mantendrá constante. Dado un conjunto de parejas definido para un weekend, por la propiedad de torneo espejado es seguro que se recorrerá dos veces la distancia entre los integrantes de cada pareja en dicho weekend o en su correspondiente en la otra mitad (en uno de los dos la pareja es local). Para los weekends interparejas, se recorrerá una vez la distancia intrapareja en cada uno.

Hemos realizado implementaciones del MCP tanto en base al Modelo 1 (versión original y versión ids) y al Modelo 3, siendo el primero, en su versión original, mucho más performante para obtener parejas fijas, mientras que el segundo funciona mejor para parejas dinámicas. Los resultados de esta comparación se pueden apreciar en la Tabla 3.1.

Instancias		Parejas Fijas			Parejas Dinámicas		
Matriz	Restricciones de localía	M 1 Orig	M 1 Ids	M3	M 1 Orig	M 1 Ids	M 3
A1M-2007-2008	no-Restr	0.48	9.18	2.01	-	-	1.33
	no-FOR-MIS	1.67	2.37	2.91	-	-	14.65
A1M-2008-2009	no-Restr	2.02	-	1.53	-	-	0.56
	no-CHU-NQN	1.13	8.38	3.01	-	-	0.80
	no-UPC-OSJ	0.63	-	6.68	-	-	2.60
A1M-2009-2010	no-Restr	0.62	-	1.91	-	-	0.43
	no-TUC	6.30	7.80	6.80	(160%)	(∞%)	(2%)
Cercanos	no-Restr	0.30	-	2.10	-	-	0.42
	no-L-G	0.26	7.23	4.95	-	-	3.62
Lejanos	no-Restr	0.10	-	2.10	-	-	0.40
	no-K-A-D	0.42	11.00	3.85	-	-	2.11
16Equipos	no-Restr	-	13.20	-	(∞%)	(∞%)	(∞%)
	conRestr	(∞%)	(∞%)	(∞%)	(∞%)	(∞%)	(15%)

Tabla 3.1: Tiempos de implementaciones de MCP (en minutos), límite 15 min

3.2.2. Modelo de parejas prefijadas y esquema de visita fijo

Como se mencionó anteriormente, los fixtures en los últimos años se realizaron calculando una variante del TTP de 6 parejas con esquema de visita

fijo. Puesto que el modelo anterior devuelve una configuración de parejas interesante bajo un criterio de cercanía, retomaremos la idea antigua y modificaremos los modelos originales para que se restrinjan a una configuración de parejas dada. Para que estos modelos puedan finalizar en un tiempo razonable, también les incorporaremos el esquema de visita fijo.

El resultado deberá ser equivalente a realizar el TTP de 6 parejas en el caso que éstas sean fijas y lo más eficiente posible en el caso dinámico. Nos referiremos a este modelo con la sigla EVF (esquema de visita fijo), y le anexaremos el prefijo PF o PD, según la configuración inicial sea de parejas fijas o dinámicas.

Definiremos previamente algunos conjuntos que serán utilizados por los modelos:

- $PF \subseteq \{(i, j) \in Equipos \times Equipos, id(i) > id(j)\}$: conjunto que especifica $\frac{n}{2}$ parejas fijas durante todo el torneo (configuración de parejas fijas).
- $PD \subseteq \{(i, j) \in Equipos \times Equipos \times Jueves, id(i) > id(j)\}$: conjunto que especifica $\frac{n}{2}$ parejas por weekend (configuración de parejas dinámicas).

A continuación presentaremos las implementaciones de EVF a partir de los distintos modelos:

Modelo 1

En ambos casos, se toma la versión original del modelo, y se anexan algunas familias de restricciones. Mostraremos la versión en base a la versión original P_{ijk} , pero se puede aplicar a cualquiera de las variantes de forma análoga.

Parejas Fijas A la modificación de parejas fijas se le agregan:

1. Parejas prefijadas:

$$P_{ij} = 1$$

$$(i, j) \in PF$$

2. Esquema de visita fijo:

$$X_{ikf} + X_{jqf} + X_{iq(f+1)} + X_{jk(f+1)} = 0$$

$$(i, j), (q, k) \in PF, i \neq q, f \in Jueves$$

3. Equivalencias de viajes Sábado-Jueves entre variables Y_{ijf} por parejas prefijadas:

$$Y_{iqf} = Y_{jkf}$$

$$(i, j), (q, k) \in PF, f \in Sabados - \{IS1, IS2\}$$

4. Equivalencias de viajes Sábado-Jueves entre variables Y_{ijf} por parejas prefijadas en los weekends interparejas:

$$Y_{iqf} = Y_{ikf}$$

$$(i, j), (q, k) \in PF, f \in \{IS1, IS2\}$$

5. Anulación de variables Y_{ijf} por esquema de visita fijo:

$$Y_{ikf} + Y_{jqf} = 0$$

$$(i, j), (q, k) \in PF, f \in Sabados$$

Parejas Dinámicas En este caso no se puede asumir nada sobre las variables Y_{ijf} , puesto que de una fecha a otra una pareja visitante puede provenir de dos parejas distintas de la fecha anterior. En este caso sólo agregamos:

1. Parejas prefijadas:

$$P_{ijf} = 1$$

$$(i, j, f) \in PD$$

2. Esquema de visita fijo:

$$X_{ikf} + X_{jqf} + X_{iq(f+1)} + X_{jk(f+1)} = 0$$

$$(i, j, f), (q, k, f) \in PD, i \neq q, f \in Jueves$$

Modelo 2

Parejas Fijas Definiremos el siguiente conjunto:

$$Primeros = \{i \setminus (i, j) \in PF\}$$

Se realizan las siguientes modificaciones al modelo original:

1. Juegan todos contra todos ((5) en el modelo original):

$$\sum_{\substack{k \in Equipos \\ f \in Fechas \\ f < n}} Y_{kijf} + \sum_{\substack{z \in Equipos \\ f \in Fechas \\ f < n}} Y_{zjif} = 1$$

$$i, j \in Primeros, i \neq j$$

2. Un equipo no juega tres weekends seguidos de local ((6) en el modelo original):

$$\left(\sum_{j \in Equipos} Y_{jkk(f-1)} \right) + Y_{kkk(f+1)} + Y_{kkk(f+3)} \leq 2$$

$$i \in Primeros, f \in Jueves1M, IJ1 < f < n - 3$$

3. Un equipo no juega tres weekends seguidos de visitante((7) en el modelo original):

$$\sum_{\substack{i,j \in \text{Equipos} \\ j \neq k}} (Y_{ijk(f-1)} + Y_{ijk(f+1)} + Y_{ijk(f+3)}) \leq 2$$

$$i \in \text{Primeros}, f \in \text{Jueves}1M, IJ1 < f < n - 3$$

4. Comportamiento de parejas prefijadas más rotación de rivales, se reemplaza (12) del modelo original por:

$$Y_{ijkf} = Y_{jiqf}$$

$$(i, j), (q, k) \in PF, i \neq q, f \in (\text{Jueves} - \{IJ1, IJ2\})$$

5. Variables anuladas por parejas prefijadas:

$$\sum_{k \in \text{Equipos}} (Y_{ijkf}) = 0$$

$$(i, j) \notin PF, (j, i) \notin PF, f \in (\text{Jueves} - \{IJ1, IJ2\})$$

6. Variables anuladas Jueves-Sábado para asegurar esquema de visita fijo:

$$Y_{ijkf} + Y_{jiqf} = 0$$

$$(i, j), (q, k) \in PF, i \neq q, f \in (\text{Jueves} - \{IJ1, IJ2\})$$

7. Variables anuladas Sábado-Jueves para asegurar esquema de visita fijo:

$$\sum_{h \in \text{Equipos}} (Y_{hik(f-1)}) + \sum_{h \in \text{Equipos}} (Y_{hjq(f-1)}) = 0$$

$$(i, j), (q, k) \in PF, i \neq q, f \in (\text{Jueves} - \{IJ1, IJ2\})$$

8. Equivalencias de viajes Sábado-Jueves entre variables Y_{ijkf} por parejas prefijadas:

$$Y_{iqzf} = Y_{ijt f}$$

$$Y_{iqtf} = Y_{jkzf}$$

$$Y_{iqtf} = Y_{jkzf}$$

$$Y_{jqzf} = Y_{ikt f}$$

$$Y_{jqtf} = Y_{ikzf}$$

$$(i, j), (q, k), (z, t) \in PF, i \neq q, f \in (\text{Sabados} - \{IS1, IS2\})$$

Parejas Dinámicas

Partiendo nuevamente del modelo original:

1. Comportamiento de parejas prefijadas más rotación de rivales, se reemplaza (12) del modelo original por:

$$Y_{ijkf} = Y_{jiqf}$$

$$(i, j, f), (q, k, f) \in PD, i \neq q, f \in (\text{Jueves} - \{IJ1, IJ2\})$$

2. Variables anuladas por parejas prefijadas:

$$\sum_{k \in Equipos} (Y_{ijkf}) = 0$$

$$(i, j, f) \notin PD, (j, i, f) \notin PD, f \in (Jueves - \{IJ1, IJ2\})$$

3. Variables anuladas Jueves-Sábado para asegurar esquema de visita fijo:

$$Y_{ijkf} + Y_{jiqf} = 0$$

$$(i, j, f), (q, k, f) \in PD, i \neq q, f \in (Jueves - \{IJ1, IJ2\})$$

4. Variables anuladas Sábado-Jueves para asegurar esquema de visita fijo:

$$\sum_{h \in Equipos} (Y_{hik(f-1)}) + \sum_{h \in Equipos} (Y_{hj(q-f-1)}) = 0$$

$$(i, j, f), (q, k, f) \in PD, i \neq q, f \in (Jueves - \{IJ1, IJ2\})$$

Modelo 3

Este modelo es el que más directamente se adapta a una configuración de parejas prefijada, de hecho, en el caso de parejas fijas es equivalente al modelo utilizado en 3.1. En ambos casos la principal modificación consiste en restringir el dominio de las variables X_{ijqkw} y Z_{ijw} .

Parejas Fijas

1. Se redefinen las variables X_{ijqkf} y Z_{ijf} :

$$X_{ijqkw} \text{ con } i, j, q, k \in Equipos, id(i) > id(j), id(q) > id(k), i \neq j \neq q \neq k, (i, j), (q, k) \in PF, w \in (Weekends - \{IW1, IW2\})$$

$$H(i, j, q, k, w) =$$

$$i, j, q, k \in Equipos, id(i) > id(j), id(q) > id(k), i \neq j \neq q \neq k, (i, j), (q, k) \in PF, w \in (Weekends - \{IW1, IW2\})$$

$$Z_{ijw} \text{ con } i, j \in Equipos, i \neq j, ((i, j) \in PF \vee (j, i) \in PF), w \in \{IW1, IW2\}$$

Se mantiene el mismo conjunto de restricciones que en el modelo original, anulando las variables que queden fuera de los nuevos dominios. En consecuencia, algunos conjuntos de restricciones se transformarán en uno solo. Un ejemplo de esto son las restricciones que piden que dos equipos que forman pareja se enfrenten con un tercero, si bien se definen para cada uno de los dos equipos, las restricciones serán iguales.

2. Equivalencias de viajes Sábado-Jueves entre variables Y_{ijf} por parejas prefijadas:

$$Y_{iqf} = Y_{jkf}$$

$$(i, j), (q, k) \in PF, f \in Sabados - \{IS1, IS2\}$$

3. Equivalencias de viajes Sábado-Jueves entre variables Y_{ijf} por parejas prefijadas en los weekends interparejas:

$$Y_{iqf} = Y_{ikf}$$

$$(i, j), (q, k) \in PF, f \in \{IS1, IS2\}$$

Estas últimas familias de restricciones, en conjunción con los dominios restringidos, hacen que también las restricciones de las variables Y_{ijk} se fusionen. Esto termina de formar la equivalencia entre este modelo y el original utilizado para resolver el problema.

Parejas Dinámicas

1. Se redefinen las variables X_{ijqkf} y Z_{ijf} :

$$X_{ijqkw} \text{ con } i, j, q, k \in \text{Equipos}, id(i) > id(j), id(q) > id(k), i \neq j \neq q \neq k, (i, j, 2w) \in PD, w \in (\text{Weekends} - \{IW1, IW2\})$$

$$H(i, j, q, k, w) =$$

$$i, j, q, k \in \text{Equipos}, id(i) > id(j), id(q) > id(k), i \neq j \neq q \neq k, (i, j, 2w), (q, k, 2w) \in PD, w \in (\text{Weekends} - \{IW1, IW2\})$$

$$Z_{ijw} \text{ con } i, j \in \text{Equipos}, i \neq j, ((i, j, 2w) \in PD \vee (j, i, 2w) \in PD), w \in \{IW1, IW2\}$$

Se mantiene el mismo conjunto de restricciones que en el modelo original, anulando las variables que queden fuera de los nuevos dominios.

Se han testado las 3 versiones del modelo de parejas prefijadas con esquema de visita fijo (tablas 3.2 y 3.3). En el caso del Modelo 1 se ha probado con la versión original y con la versión ids. Tanto con la configuración de parejas fijas como con la configuración de parejas dinámicas, el Modelo 2 obtuvo los mejores tiempos. Las restricciones de localía tienen un peso muy grande en este caso, las instancias sin restricciones de este tipo demoran considerablemente más tiempo en resolver que las demás.

Resulta sencillo relajar cualquiera de estos modelos para que contemplen un esquema de visita dinámico (EVD), no obstante, la performance desmejora notablemente en especial en los casos donde no hay restricciones de localía (tablas 3.4 y 3.5).

Una vez resuelto este modelo, se obtiene en un tiempo aceptable un fixture de buena calidad, capaz de respetar los estándares actuales de la Liga de Voleibol en el caso de parejas fijas.

De aquí en adelante no intentaremos modificar los apareamientos de equipos, pero aún queda pendiente optimizar en base al esquema de visita que estuvo fijo o fue insignificante en los modelos anteriores.

Instancia	Restricciones de localía	M 1 Orig	M 1 Ids	M2	M 3
A1M-2007-2008	no-Restr	118.83	-	13.75	42.95
	no-FOR-MIS	5.73	12.17	2.13	2.22
A1M-2008-2009	no-Restr	90.35	-	24.25	29.00
	no-CHU-NQN	27.93	47.18	4.00	8.68
	no-UPC-OSJ	31.85	42.65	4.93	9.30
A1M-2009-2010	no-Restr	117.82	-	14.10	65.65
	no-TUC	2.50	5.83	1.17	0.88
Cercanos	no-Restr	57.35	-	19.31	77.80
	no-L-G	2.38	4.62	1.50	0.73
Lejanos	no-Restr	73.60	101.13	19.03	16.18
	no-K-A-D	4.90	9.75	2.10	2.67
16Equipos	no-Restr	(234 %)	(282 %)	(47 %)	(216 %)
	conRestr	(155 %)	(198 %)	(17 %)	(141 %)

Tabla 3.2: Comparación de tiempos (minutos) parejas prefijadas esquema de visita fijo, parejas fijas (PF-EVF), límite 2hs

Instancia	Restricciones de localía	M 1 Orig	M 1 Ids	M2	M 3
A1M-2007-2008	no-Restr	-	-	117.77	-
	no-FOR-MIS	2.78	2.28	3.40	2.17
A1M-2008-2009	no-Restr	(48 %)	(35 %)	(6 %)	(21 %)
	no-CHU-NQN	6.38	13.52	6.83	4.78
	no-UPC-OSJ	11.43	19.60	8.65	12.31
A1M-2009-2010	no-Restr	-	-	44.90	-
	no-TUC	0.40	0.87	1.50	0.22
Cercanos	no-Restr	114.45	-	-	-
	no-L-G	0.18	0.33	0.13	0.06
Lejanos	no-Restr	92.00	-	-	-
	no-K-A-D	1.00	1.50	3.50	0.67
16Equipos	no-Restr	(248 %)	(283 %)	(62 %)	(209 %)
	conRestr	(162 %)	(196 %)	(17 %)	(142 %)

Tabla 3.3: Comparación de tiempos (minutos) de implementaciones de parejas prefijadas esquema de visita fijo, parejas dinámicas (PD-EVF), límite 2hs

Instancia	Restricciones de localía	M 1 Orig	M 1 Ids	M2	M 3
A1M-2007-2008	no-Restr	(155 %)	(273 %)	(13 %)	(143 %)
	no-FOR-MIS	-	-	7.27	-
A1M-2008-2009	no-Restr	(131 %)	(196 %)	(11 %)	(132 %)
	no-CHU-NQN	-	-	19.20	-
	no-UPC-OSJ	-	-	37.90	-
A1M-2009-2010	no-Restr	(112 %)	(186 %)	(8 %)	(107 %)
	no-TUC	-	-	4.83	-
Cercanos	no-Restr	(122 %)	(186 %)	(13 %)	(115 %)
	no-L-G	-	-	7.03	-
Lejanos	no-Restr	(140 %)	(219 %)	(21 %)	(122 %)
	no-K-A-D	-	-	11.80	-
16Equipos	no-Restr	(243 %)	(∞ %)	(∞ %)	(261 %)
	conRestr	(196 %)	(∞ %)	(40 %)	(197 %)

Tabla 3.4: Comparación de tiempos (minutos) de implementaciones de parejas prefijadas esquema de visita dinámico, parejas fijas (PF-EVD), limite 2hs

Instancia	Restricciones de localía	M 1 Orig	M 1 Ids	M2	M 3
A1M-2007-2008	no-Restr	(187 %)	(∞ %)	(24 %)	(172 %)
	no-FOR-MIS	-	-	40.30	-
A1M-2008-2009	no-Restr	(144 %)	(191 %)	(34 %)	(155 %)
	no-CHU-NQN	-	-	93.70	-
	no-UPC-OSJ	(82 %)	(∞ %)	(9 %)	(93 %)
A1M-2009-2010	no-Restr	(124 %)	(∞ %)	(24 %)	(113 %)
	no-TUC	4.32	33.45	11.15	2.73
Cercanos	no-Restr	(132 %)	(∞ %)	(25 %)	(134 %)
	no-L-G	16.63	-	5.67	1.50
Lejanos	no-Restr	(152 %)	(∞ %)	(43 %)	(141 %)
	no-K-A-D	-	-	23.17	-
16Equipos	no-Restr	(274 %)	(∞ %)	(∞ %)	(238 %)
	conRestr	(206 %)	(∞ %)	(∞ %)	(185 %)

Tabla 3.5: Comparación de tiempos (minutos) de implementaciones de parejas prefijadas esquema de visita dinámico, parejas dinámicas (PD-EVD), limite 2hs

3.2.3. Esquema de visita dinámico: metaheurística en base a programación entera

La solución obtenida por los modelos anteriores es buena, sin embargo, se pierden posibilidades de mejora al considerar un esquema de visita fijo. Si se procesa la solución con un modelo que sólo permita relajar el esquema de visita, es muy posible que se note alguna mejora, sin embargo, ello no garantiza que se tenga el óptimo para esa configuración de parejas con esquema de visita dinámico.

En esta parte del trabajo propondremos versiones con modelos de programación entera de las búsquedas locales de la Sección 1.3. En base a estos modelos construiremos a su vez una metaheurística Tabu Search para mejorar la solución obtenida hasta el momento, incorporando el esquema de visita dinámico.

Implementación de las búsquedas locales con programación entera

Las búsquedas locales mencionadas en la Sección 1.3 no son extensivas, sino que sólo consideran una pequeña gama de movimientos a efectos de recorrer un espacio acotado. En este trabajo nos proponemos reimplementar estas búsquedas locales de forma extensiva con modelos de programación entera a efectos de analizar si es factible utilizarlas en esta modalidad. La idea es partir de los resultados acumulados de los dos modelos anteriores, es decir, mantener la configuración de parejas del modelo de la Sección 3.2.1 y usar como solución inicial la del modelo de parejas prefijadas de la Sección 3.2.2.

HAP de parejas prefijado

Dada una solución válida para una instancia del problema, se le puede extraer su HAP por parejas, es decir la especificación de qué parejas son locales y visitantes en cada weekend (excepto el interparejas donde serían local/visitante a la vez). Luego se puede reoptimizar fijando HAP, reordenando los enfrentamientos entre locales y visitantes.

A diferencia de los modelos de las secciones anteriores, en esta búsqueda local incorporaremos el concepto de esquema de visita dinámico. Así, cada posible encuentro entre una pareja local y una visitante en una fecha tendrá dos versiones distintas.

En principio a este modelo podría permitírsele generar cambios en el esquema de parejas (juntando dos equipos locales/visitantes que no eran pareja originalmente). Sin embargo, el costo en tiempo entre resolver este modelo por HAP de equipos en vez de HAP de parejas es muy alto. Aparte, es muy poco probable que el modelo encuentre una mejor solución a partir de cambios entre parejas, ya que deben darse condiciones muy fuertes entre los patrones de localía de los equipos involucrados. Una propiedad a recalcar es que en el caso de parejas fijas, esos cambios directamente no son posibles. Dos

equipos que formen una pareja fija compartirán un mismo patrón de localía a lo largo del fixture, esto implica que si se desea romper la pareja, necesariamente deberá existir un tercer equipo con el mismo patrón. Pero esto último no es posible puesto que el tercer equipo debe enfrentarse en algún momento a la pareja original, por lo que habrá una fecha donde juegue de visitante y los otros de local, o al revés.

En base a cualquiera de los tres modelos originales, hay básicamente dos formas de representar un nuevo modelo que represente al HAP de parejas prefijado:

Si se sabe que el equipo A juega de visitante en la fecha f se puede encarar la situación de distintas maneras:

1. Se pueden agregar restricciones para que se cumpla alguna de todas las posibilidades en que efectivamente A juegue de visitante en f .
2. Se pueden prohibir todos los casos donde A juegue de local en esa fecha, esto puede lograrse con restricciones o delimitando los dominios de las variables.

Las dos formas son equivalentes, en particular optaremos por reducir los dominios de las variables ya que es la que más acota los modelos y los hace más fácil de generar/procesar. Como veremos más adelante, nos interesará generar y resolver numerosas instancias de este modelo, por lo que la eficiencia en este apartado resulta importante.

Para implementar este modelo, tomamos cualquiera de los de la Sección 3.2.2 (versión parejas fijas o dinámicas) y restringiremos el dominio de sus variables según corresponda. Para permitir el esquema de visita dinámico, en los modelos 1 y 2 se eliminan las restricciones pertinentes, y en el 3 se relaja el dominio en este aspecto. A su vez, en los 3 modelos se eliminan las restricciones de tramo, ya que se asume que al momento de generarlos tienen como entrada un HAP válido que ya las cumpla.

Las pruebas que hemos realizado demostraron que el modelo más eficiente para esta búsqueda local es la versión en base al Modelo 2.

Intercambio de rivalidades entre equipos

Partiendo del Modelo 3:

- Se parte de una solución inicial S
- Se eliminan las restricciones de las variables X_{ijqkw} de todos contra todos y un sólo partido por fecha. Permanecen las restricciones de tramo y torneo espejado.

- Se agregan las siguientes restricciones, para garantizar que los partidos de la solución inicial se jueguen en el orden original o espejados:

$$X_{ijqkw} + X_{qkij(w+n/2)} = 1$$

$$i, j, q, k \in \text{Equipos}, w \in (\text{Weekends1M} - \{IW1\}),$$

$$H(i, j, q, k, w), id(q) > id(k), X_{ijqkw} \in S$$

$$X_{ijqkw} + X_{kqji(w+n/2)} = 1$$

$$i, j, q, k \in \text{Equipos}, w \in (\text{Weekends1M} - \{IW1\}),$$

$$H(i, j, q, k, w), id(q) < id(k), X_{ijqkw} \in S$$

- Las restricciones pertinentes a las variables Y_{ijw} quedan iguales.

Es posible implementar esto en base al Modelo 1, pero resulta mucho menos eficiente. Esto ocurre porque en este modelo, un encuentro entre parejas en un weekend está definido por cuatro variables X_{ijf} , por lo que invertir una de estas variables no es correcto si no se invierten de forma acorde las otras tres. Si se intenta simplificar esto para que sólo se tome en cuenta una de las cuatro variables, se llega a la versión del Modelo 3.

Esta vecindad es la que más rápido resuelve, ya que cuenta con un número muy reducido de variables, y varias de las restricciones más importantes en los modelos originales se eliminan (“todos contra todos” y “un partido por fecha”).

Esta vez no hay concepto de esquema de visita, puesto que la solución original lo impone, el modelo sólo puede invertir los partidos.

Hemos descartado realizar este vecindario en base al Modelo 2, esto se debe a que las variables Y_{ijkf} de este modelo deben especificar de dónde viene el equipo visitante, lo que complica las restricciones. Esta situación no se da con los otros dos modelos ya que la información está más distribuida entre las distintas variables.

Por lo dicho anteriormente, la implementación en base al Modelo 3 es la preferida.

Intercambio de fechas

Dado un fixture válido, si se intenta reordenar las fechas, sólo se modificarán las distancias recorridas entre Sábado y Jueves, ya que las distancias Jueves-Sábado se mantienen fijas. Luego, podemos pensar que cada weekend tiene un peso fijo $p(w)$ asociado, conformado por la suma de los viajes Jueves-Sábado que se realizan dentro de cada uno.

Sabiendo los partidos que se juegan en cada weekend, es posible armar una matriz de distancias Sábado-Jueves, donde $dist_{w_1w_2}$ representa la suma de los recorridos de los equipos que se trasladan del Sábado de w_1 al Jueves de w_2 . La matriz no resulta simétrica debido a que los equipos no comienzan y terminan el weekend en el mismo lugar (se desplazan entre Jueves y Sábado)

A partir de estos datos se puede plantear un modelo donde haya que ordenar los distintos weekends en una secuencia de slots para llenar el fixture. La función objetivo contemplará sólo los $dist_{w_1w_2}$, y no los $p(w)$ ya que son un valor fijo en el problema. Igualmente, pueden agregarse como constantes en la función objetivo, a efectos de lograr mayor claridad en la solución.

Este modelo será implementado por weekends y no por fechas, en principio por cuestiones de eficiencia, pero también porque es muy difícil que se dé un caso que permita reordenar las fechas individuales de forma que quede un esquema de parejas válido.

Los weekends interparejas sólo podrán intercambiarse entre sí.

Modelo de intercambio de weekends

Definiciones:

- $dist_{w_1w_2}$: matriz de distancia entre los weekends w_1 y w_2 (no simétrica).
- hap_{we} : matriz de $Weekends \times Equipos$ que vale 0 si el equipo e es local o 1 si es visitante en el weekend w .
- $Slots$: conjunto de posiciones a ocupar por los weekends en el fixture. Se agrega uno más (slot $n + 1$ en este caso) al conjunto para marcar el regreso a la ciudad de los equipos al terminar el torneo.
- $Weekend 0$: elemento extra agregado al conjunto de Weekends, representa un weekend donde todos los equipos se mantienen en sus ciudades Jueves y Sábado. Se utiliza para marcar la vuelta a casa al final del torneo.

Variables:

- R_{ij} con $i \in Weekends, j \in Slots$
 $R_{ij} = 1$ si el slot j es ocupado por el weekend i .
- Y_{kij} con $i, j \in Weekends, i \neq j, k \in Slots$
 $Y_{kij} = 1$ si en el slot k se encuentra el weekend i , y en el slot $k + 1$ el weekend j .

Función objetivo:

$$\text{MIN} \sum_{\substack{i, j \in Weekends \\ k \in Slots}} dist_{ij} * Y_{kij}$$

Restricciones:

1. Weekends interparejas restringidos y regreso al final del torneo:

$$R_{IW1, IW1} + R_{IW2, IW1} = 1$$

$$R_{IW1, IW2} + R_{IW2, IW2} = 1$$

$$R_{0, n+1} = 1$$

2. Un weekend por slot:

$$\sum_{\substack{w \in Weekends \\ s \in Slots}} R_{ws} = 1$$

3. Un slot por weekend:

$$\sum_{s \in Slots} R_{ws} = 1 \\ w \in Weekends$$

4. Torneo espejado (1): los dos weekends correspondientes se mantienen en sus respectivas mitades del torneo:

$$R_{ws} = R_{(w+n/2)(s+n/2)} \\ s \in Slots, s < IW2, w \in Weekends, w < IW2$$

5. Torneo espejado (2): los dos weekends correspondientes se invierten en el torneo:

$$R_{ws} = R_{(w-n/2)(s+n/2)} \\ s \in Slots, s < IW2, w \in Weekends, w \geq IW2$$

6. Seteo de las variables Y_{kij} :

$$Y_{kij} \geq R_{ik} + R_{j(k+1)} - 1 \\ s \in Slots, i, j \in Weekends$$

Restricciones de localía:

7. El equipo i debe jugar de local en el weekend w :

$$R_{ws} = 0 \\ i \in Equipos, w \in Weekends, s \in Slots, \\ s = w, (i, w) \in RWLocal, i \text{ juega de visitante en } w$$

8. El equipo i debe jugar de visitante en el weekend w :

$$R_{ws} = 0 \\ i \in Equipos, w \in Weekends, s \in Slots, \\ s = w, (i, w) \in RWVisitante, i \text{ juega de local en } w$$

9. No hay equipos que jueguen 3 weekends seguidos de local:

$$\sum_{\substack{w \in Weekends \\ hap_{we}=0}} (R_{ws} + R_{w(s+1)} + R_{w(s+2)}) \leq 2 \\ s \in Slots, IW1 < s < n/2 - 1, e \in Equipos$$

10. No hay equipos que jueguen 3 weekends seguidos de visitante:

$$\sum_{\substack{w \in \text{Weekends} \\ \text{hap}_{we}=1}} (R_{ws} + R_{w(s+1)} + R_{w(s+2)}) \leq 2$$

$$s \in \text{Slots}, IW1 < s < n/2 - 1, e \in \text{Equipos}$$

Una vez se obtiene una solución, una variables R_{ws} significará que los partidos del weekend w en el fixture original se jugarán en el weekend s del nuevo.

Intercambio de equipos

Abstrayendo las posiciones de los equipos en el fixture de una solución dada, se obtiene un conjunto de n slots a ser ocupados por los n equipos. En este caso, quitando conflictos por restricciones de localía, cualquier forma de llenar los slots por los equipos generará un fixture válido. El siguiente modelo, generado a partir de una solución inicial, definirá una asignación de equipos a slots para minimizar las distancias. Este problema es un caso particular del problema de asignación cuadrática (ver [3]):

Modelo de intercambio de equipos

Definiciones:

- $\text{Slots} \equiv \{1 \dots n\}$: conjunto de posiciones a ocupar por los equipos en el fixture.
- $\text{coefs}(s_1, s_2)$: en el fixture original, cuántas veces se recorre la distancia entre los slots s_1 y s_2 (en ambos sentidos).

Variables:

- Q_{ij} con $i \in \text{Equipos}, j \in \text{Slots}$
 $Q_{ij} = 1$ si el slot j es ocupado por el equipo i .
- Y_{ijkl} con $i, k \in \text{Equipos}, j, l \in \text{Slots } j < l$
 $Y_{ijkl} = 1$ sii $Q_{ik} = 1$ y $Q_{jl} = 1$.

Función objetivo:

$$\text{MIN} \sum_{\substack{j, l \in \text{Slots} \\ j < l \\ k \in \text{Fechas}}} \text{coefs}(i, j) * Y_{ijk}$$

Restricciones:

1. Un equipo por slot:

$$\sum_{e \in \text{Equipos}} Q_{es} = 1$$

$$s \in \text{Slots}$$

2. Un slot por equipo:

$$\sum_{s \in Slots} Q_{es} = 1$$

$$e \in Equipos$$

3. Correspondencia variables Y con variables Q (1);

$$\sum_{j \in Equipos} Y_{jsit} = Q_{it}$$

$$s, t \in Slots, i \in Equipos$$

4. Correspondencia variables Y con variables Q (2);

$$\sum_{s \in Slots} Y_{jsit} = Q_{it}$$

$$t \in Slots, i, j \in Equipos$$

5. Equivalencia entre variables Y :

$$Y_{jsit} = Y_{itjs}$$

$$s, t \in Slots, i, j \in Equipos$$

Restricciones de localía:

6. El equipo i debe jugar de local en el weekend w :

$$Q_{is} = 0$$

$$(i, w) \in RWLocal$$

7. El equipo i debe jugar de visitante en el weekend w :

$$Q_{is} = 0$$

$$(i, w) \in RWVisitante$$

Metaheurística Tabu Search con programación entera

Hay una relación aprovechable entre las búsquedas locales de HAP de parejas prefijado y la de intercambio de rivalidades entre equipos. Esta se trata de una mutua exclusión entre las distintas vecindades, ya que la primera busca vecinos sin modificar el HAP mientras que la otra sólo considera vecinos con un HAP distinto.

Siempre y cuando generen mejoras a la solución cada vez que se las utilice, se pueden correr estas búsquedas locales alternadamente para mejorar una solución inicial. Es decir, podemos definir el siguiente algoritmo:

```

Heuristica(SolucionInicial)
{
  Mientras haya mejora
  {
    Obtener el HAP de la mejor solución
    Mejorar la solución con vecindario de HAP de parejas prefijadas
    Mejorar la solución con vecindario de intercambio de rivalidades
  }
}

```

Ahora bien, con este algoritmo, siempre se llegará al punto donde ninguna de las dos búsquedas locales pueda mejorar la solución, esto puede darse porque se llegó al óptimo del problema (muy poco probable), o porque se llegó a un mínimo local compartido entre las vecindades (altamente probable). En ese momento los modelos pueden devolver dos cosas:

1. La misma solución de la que se parte.
2. Una solución distinta a la inicial pero equivalente en peso.

Lo más conveniente es que los modelos devuelvan la segunda posibilidad, ya que por la propiedad de mutua exclusión, el modelo que se ejecute a continuación dispondrá de una vecindad nueva para explorar. Para forzar esta situación, extenderemos los modelos. Se les agregará a ambos una nueva restricción, la cual prohibirá que se repita la solución original de la que se parte. Dependiendo del modelo de base, esta restricción tendrá la siguiente forma general:

$$\sum_{v \in S_0} v \leq \#S_0 - 1$$

Donde:

- *Variables* es el conjunto completo de las variables del modelo
- $S_0 = \{v \in Variables / v \neq 0 \text{ en la solución anterior} \}$

Si la solución inicial era un óptimo local en la vecindad correspondiente a alguno de los modelos, el modelo devolverá una solución distinta, pero peor que la solución inicial. En principio esto no tendría porqué ser un problema mayor, ya que debería abrir una puerta para explorar un espacio de soluciones mayor. Sin embargo, el modelo de rivalidades bien podría volver a un HAP antiguo, ya recorrido antes de llegar a la solución que se le pasó por parámetro, provocando un ciclo en la heurística.

Para solucionar esto tomaremos ventaja de la eficiencia y velocidad del modelo de intercambio de rivalidades. Cada vez que se use el modelo de HAP

prefijado, se guardará el HAP en una lista Tabu. Luego, a la manera de la restricción que se agregó antes, se incluirá una nueva restricción por cada HAP en la lista, prohibiéndolo. La velocidad de resolución de este modelo hace que pueda soportar varios cientos de restricciones de HAP sin pérdida considerable de tiempo. De esta forma, la metaheurística irá iterando entre HAPs buscando mejorar la solución.

Puesto que ahora se les permite “empeorar” a las soluciones, la metaheurística iterará un número arbitrario de veces definido por el usuario.

```

HeuristicaTabu(SolucionInicial, iteraciones)
{
  Crear una lista tabú vacía
  Repetir (iteraciones) veces
  {
    Obtener el HAP de la mejor solución
    Mejorar la solución con vecindario de HAP (restringido)
    Agregar el HAP de la nueva solución a la lista tabú
    Mejorar la solución con vecindario de intercambio de rivalidades
    (restringida con la lista tabú)
  }
}

```

Nos referiremos a este algoritmo con la sigla TSPE (Tabu Search, Programación Entera).

La principal diferencia entre lo que se hace en esta metaheurística contra lo que hace el modelo de parejas predefinidas es el hecho de que el modelo de HAP prefijado puede jugar con un esquema de visita dinámico, llegando a soluciones que no eran contempladas.

Adicionalmente, hemos experimentado intercalar las dos búsquedas locales restantes en nuestro algoritmo. Puesto que son menos performantes, decidimos usarlas cada 5 iteraciones, llamaremos a esta versión de la metaheurística TSPE-IF-IE (TSPE + Intercambio de fechas + Intercambio de Equipos). El resultado en todas nuestras pruebas fue que tanto el intercambio de equipos como el intercambio de fechas devuelven siempre la solución original, sin mejora. Por lo tanto, decidimos experimentar restringiendo a estos dos modelos para que no devuelvan la solución original, para que al menos introduzcan varianza para las demás búsquedas. En ambos casos los modelos bajaron notablemente su performance. En el caso particular del intercambio de fechas, el resultado fue que las siguientes soluciones a la óptima (la original) resultaban ser mucho peores, por lo que descartamos restringir este modelo. Finalmente, en un caso hemos obtenido mejora utilizando el intercambio de equipos.

Capítulo 4

Resultados

En este capítulo expondremos los resultados obtenidos a lo largo de este trabajo.

Implementación de los modelos

Los modelos fueron codificados con el lenguaje Zimpl [13]. Este lenguaje resulta altamente flexible a la hora de definir y/o modificar modelos complejos como los que se usan en este trabajo.

Como solver se usó la utilidad SCIP [1]. SCIP es un solver de programación lógica que trabaja en conjunto con motores de solvers de programación lineal, y para los experimentos de este capítulo usamos la versión que utiliza SoPlex [26] para estos fines.

La metaheurística fue programada con scripts de bash y Perl, estos últimos se utilizaron para generar el traspaso de datos entre los modelos.

Las pruebas se hicieron en una pc Intel Core 2 Quad Q8400.

Performance de los modelos iniciales

En la Tabla 4.1 exponemos los resultados de corridas de 6 horas de los modelos iniciales (tal cual fueron presentados en el Capítulo 3) en algunas instancias. Estos modelos plantean el problema completo de obtener un fixture óptimo contemplando parejas dinámicas y esquema de visita dinámico. Se aprecia claramente que no es viable utilizar los modelos de esta manera. No obstante, resulta curioso que el Modelo 3, más pesado en cuanto a variables y restricciones, haya sido el más performante (relativamente).

Performance de la metaheurística TSPE

En la Tabla 4.2 mostramos los tiempos de nuestra metaheurística en las distintas instancias. En base a estos resultados consideramos que realizar 200 iteraciones de la metaheurística es aceptable.

En la Tabla 4.3 mostramos la iteración de la metaheurística en que se encontró la mejor solución para la Tabla 4.7. En la mayoría de los casos, alcanza con una sola iteración de TSPE para llegar al óptimo, no obstante algunos casos certifican que es beneficioso iterar una mayor cantidad de veces. En base a estos resultados, consideraremos que realizar una sola iteración de las búsquedas locales de la metaheurística es una buena aproximación del

Matriz	Restricciones de localía	M 1 Orig	M 1 Ids	M2	M 3
A1M-2007-2008	no-Restr	(∞ %)	(∞ %)	(∞ %)	(266 %)
A1M-2008-2009	no-CHU-NQN	(∞ %)	(∞ %)	(∞ %)	(214 %)
A1M-2009-2010	no-TUC	(∞ %)	(∞ %)	(∞ %)	(211 %)
Cercanos	no-L-G	(∞ %)	(∞ %)	(∞ %)	(225 %)
Lejanos	no-Restr	(∞ %)	(∞ %)	(∞ %)	(235 %)

Tabla 4.1: Resultados de los modelos iniciales (resolución del problema completo) en algunas instancias, límite de tiempo 6 hs.

Instancias		Tiempo	
Matriz	Restricciones de localía	Parejas Fijas	Parejas Dinámicas
A1M-2007-2008	no-Restr	19.48	26.4
	no-FOR-MIS	14.65	13.92
A1M-2008-2009	no-Restr	21.78	28.30
	no-CHU-NQN	14.92	15.92
	no-UPC-OSJ	16.92	16.13
A1M-2009-2010	no-Restr	19.90	25.68
	no-TUC	14.62	13.85
Cercanos	no-Restr	19.35	26.22
	no-L-G	14.50	13.95
Lejanos	no-Restr	18.50	23.22
	no-K-A-D	14.53	14.13
16Equipos	no-Restr	92.06	124.67
	conRestr	70.87	72.75

Tabla 4.2: Tiempos (minutos) de TSPE con 200 iteraciones

resultado final de iterar muchas veces (pero siempre teniendo en cuenta que existe una posibilidad de mejora si se iterara más veces).

En la Tabla 4.4 se encuentran los resultados destacados al utilizar TSPE-IF-IE (metaheurística TSPE con el agregado de los dos vecindarios restantes). En las demás instancias el resultado no mejoró, y en el caso particular de las instancias de 16 equipos las búsquedas locales agregadas no lograron terminar en un tiempo aceptable. Si bien las mejoras son notables, también es cierto que en uno de los casos la configuración de parejas de la solución final es considerablemente peor al óptimo. En el caso de una aplicación real, este dato no es menor, puesto que se incrementa la distancia que se viaja en el fragmento más pequeño de la semana.

Instancias		Parejas Fijas	Parejas Dinámicas
Matriz	Restricciones de localía	Iteración mejor solución	Iteración mejor solución
A1M-2007-2008	no-Restr	1	1
	no-FOR-MIS	1	1
A1M-2008-2009	no-Restr	1	1
	no-CHU-NQN	1	1
	no-UPC-OSJ	1	93
A1M-2009-2010	no-Restr	48	48
	no-TUC	1	1
Cercanos	no-Restr	2	2
	no-L-G	1	1
Lejanos	no-Restr	1	1
	no-K-A-D	1	1
16Equipos	no-Restr	8	1
	conRestr	1	1

Tabla 4.3: Iteración de TSPE donde se encuentra la mejor solución (entre 200 iteraciones)

Matriz	Restricciones de localía	CP óptima	EVF + TSPE 200it	EVF + TSPE C. 50 it	CP obtenida
A1M-2008-2009	no-UPC-OSJ (parejas dinámicas)	29712	125777	123964	33190
A1M-2009-2010	no-Restr	37476	131602	130835	37518

Tabla 4.4: Casos destacados donde utilizar TSPE-IF-IE obtiene resultados favorables en comparación a TSPE. Puesto que la metaheurística baja la performance, se hacen sólo 50 iteraciones. En la última columna se muestra el peso de la configuración de parejas obtenida, ya que puede variar por las búsquedas locales agregadas.

Comparación entre utilizar esquema de visita fijo y luego la metaheurística, o utilizar el esquema de visita dinámico

En las tablas 4.5 y 4.6, comparamos cuán efectivo es utilizar el esquema de visita fijo junto con la metaheurística (EVF + TSPE), en lugar de utilizar directamente un modelo que resuelva directamente con esquema de visita dinámico (EVD). En las instancias sin restricciones se descarta el caso de parejas dinámicas por ser idéntico al de parejas fijas.

Instancias		Parejas Fijas		Parejas Dinámicas	
Matriz	Restricciones de localía	EVF + TSPE 200it	EVD	EVF + TSPE 200it	EVD
A1M-2007-2008	no-Restr	107368	107368	-	-
	no-FOR-MIS	115022	114664	115459	115459
A1M-2008-2009	no-Restr	123710	123008	-	-
	no-CHU-NQN	131852	131611	127272	127272
	no-UPC-OSJ	124645	124528	125777	125777
A1M-2009-2010	no-Restr	131602	130835	-	-
	no-TUC	136300	136300	143082	143082
Cercanos	no-Restr	52141	52141	-	-
	no-L-G	56295	55418	55874	55874
Lejanos	no-Restr	160585	160530	-	-
	no-K-A-D	175766	175478	168096	168096
16Equipos	no-Restr	281750	299966	262463	279715
	conRestr	287581	299714	254920	272701

Tabla 4.5: Comparación resultados: en base a una configuración de parejas obtenida previamente por MCP, se compara entre continuar el cálculo usando el modelo de parejas prefijadas con esquema de visita fijo (EVF) y la metaheurística (TSPE) con 200 iteraciones, contra utilizar parejas prefijadas con esquema de visita dinámico (EVD). Límites de tiempo : EVF (2hs), EVD(8hs).

En lo que a parejas fijas respecta, el modelo EVD suele sacar una ventaja no demasiado grande. En el caso dinámico en general, la situación se invierte ya que nuestro procedimiento equipara al esquema dinámico. Es importante recalcar también que nuestro procedimiento superó con creces al esquema dinámico en lo que respecta a las instancias de 16 equipos, en las cuales los límites de tiempo resultan obligatorios a la hora de obtener soluciones en un lapso aceptable (EVD no logra superar nuestro método aún con 8 horas).

Instancias		Parejas Fijas		Parejas Dinámicas	
Matriz	Restricciones de localía	EVF + TSPE 200it	EVD	EVF + TSPE 200it	EVD
A1M-2007-2008	no-Restr	33.23	164.00	-	-
	no-FOR-MIS	16.78	7.27	16.09	40.30
A1M-2008-2009	no-Restr	46.03	144.03	-	-
	no-CHU-NQN	18.92	19.20	20.70	93.70
	no-UPC-OSJ	21.85	37.90	24.78	128.27
A1M-2009-2010	no-Restr	34.00	122.56	-	-
	no-TUC	15.50	4.83	15.35	2.73
Cercanos	no-Restr	38.66	144.37	-	-
	no-L-G	15.23	7.03	14.01	1.50
Lejanos	no-Restr	34.68	294.00	-	-
	no-K-A-D	16.63	11.80	14.80	23.17
16Equipos	no-Restr	224.06	480.00	256.67	480.00
	conRestr	202.87	480.00	204.75	480.00

Tabla 4.6: Comparación de los tiempos (minutos) de los resultados de la Tabla 4.5.

Resultados de nuestro esquema algorítmico

En la Tabla 4.7 resumimos las etapas del procedimiento propuesto en el capítulo anterior.

A modo de observación, en las instancias sin restricciones de localía, la configuración de parejas óptima será la mismo tanto para parejas fijas como dinámicas

En algunas instancias, se puede ver que para el caso dinámico, el óptimo del MCP no necesariamente resultará en un buen fixture al final, inclusive puede ser considerablemente peor al de parejas fijas.

Instancias		Parejas Fijas			Parejas Dinámicas		
Liga	Restricciones de localía	MCP	EVF	TSPE	MCP	EVF	TSPE
A1M-2007-2008	no-Restr	20196	107879	107368	20196	107879	107368
	no-FOR-MIS	31332	117529	115022	27620	118985	115459
A1M-2008-2009	no-Restr	29184	123738	123710	29184	123738	123710
	no-CHU-NQN	39372	133265	131852	32580	127698	127272
	no-UPC-OSJ	30768	125023	124645	29712	126805	125777
A1M-2009-2010	no-Restr	37476	131794	131602	37476	131794	131602
	no-TUC	43872	136360	136300	39522	144469	143082
Cercanos	no-Restr	14688	53022	52141	14688	53022	52141
	no-L-G	18036	56605	56295	15768	56384	55874
Lejanos	no-Restr	38448	161924	160585	38448	161924	160585
	no-K-A-D	56784	176880	175766	41342	171153	168097
16Equipos	no-Restr	63984	293533	281750	63984	267991	262463
	conRestr	63984	289128	287581	63984	261493	254920

Tabla 4.7: Resultados obtenidos en las distintas etapas del esquema algorítmico propuesto: modelo de configuración de parejas (MCP) + parejas prefijadas con esquema de visita fijo (EVF) + metaheurística (TSPE) con 200 iteraciones. En las instancias de 16 equipos, EVF tiene límite de tiempo de 2hs.

Experimentos con las configuraciones de parejas

Para mejorar estos casos, decidimos probar nuestro procedimiento con distintas configuraciones de parejas, no sólo con la óptima.

En la Tabla 4.8 podemos ver la diferencia que distintas configuraciones de parejas pueden generar sobre la solución final de una misma instancia. En la Figura 4.1 hemos graficado la distancia total recorrida entre todas las fechas en función de las distancias recorridas de Jueves a Sábado (cantidad que minimizamos al buscar una configuración de parejas). Observamos que pareciera no haber correlación alguna entre el peso de la configuración de parejas y el del fixture final. Notar que, si bien no entra en el rango del gráfico, la última configuración de la Tabla 4.8 es la más pesada y a la vez se corresponde con un fixture considerablemente bueno.

Decidimos extender este experimento a todas las instancias. Para cada instancia obtuvimos 50 configuraciones de parejas (a través del MCP), luego se utilizó el modelo de esquema de visita fijo, y finalmente una sola iteración de la metaheurística (que, como hemos visto, suele ser suficiente). En la Tabla 4.9 comparamos los resultados obtenidos por nuestro procedimiento original, contra la mejor configuración encontrada en el último experimento.

CP	EVF + HAP
39522	139522
39522	143082
39608	135957
40448	149164
39882	140146
43872	136300

Tabla 4.8: Comparación EVF + 1 iteración de búsqueda de HAP prefijado, con distintas configuraciones de parejas para la instancia A1M-2009-2010 NO-TUC con parejas dinámicas.

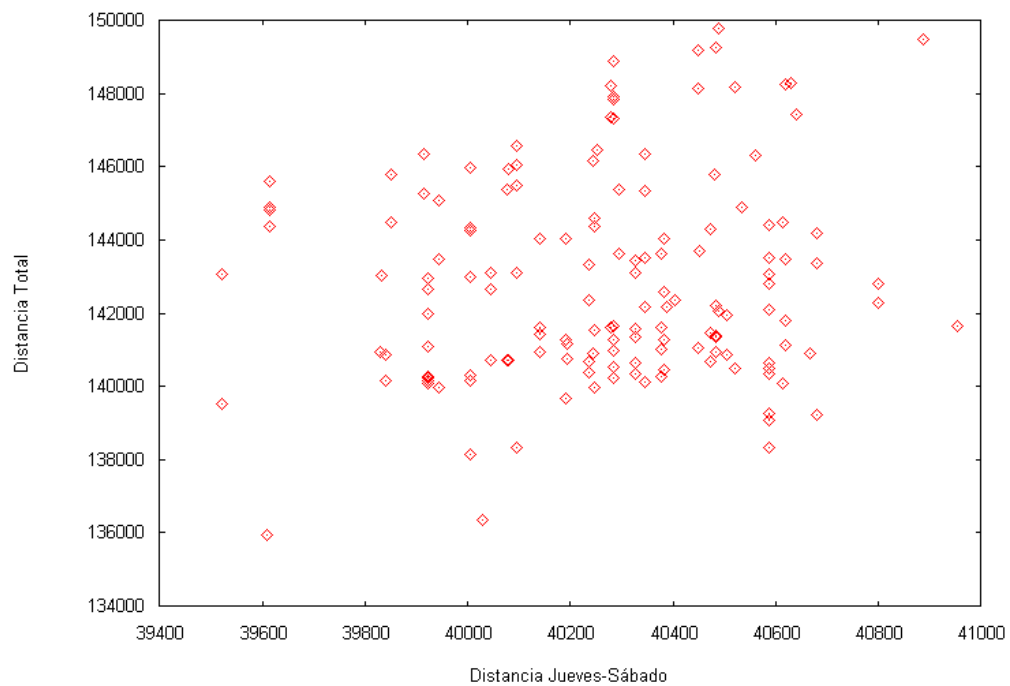


Figura 4.1: Relación entre los pesos de configuraciones de parejas y viajes totales en la instancia A1M 2009-2010 NO-TUC

Matriz	Restricciones de localía	CP óptima	PD-EVF + TSPE 200it	Mejor CP	PD-EVF + TSPE 1it
A1M-2007-2008	no-Restr	20196	107368	20196	107368
	no-FOR-MIS	27620	115459	27620	115459
A1M-2008-2009	no-Restr	29184	123710	30934	122798
	no-CHU-NQN	32580	127272	32580	127272
	no-UPC-OSJ	29712	125777	30240	124525
A1M-2009-2010	no-Restr	37476	131602	37476	131642
	no-TUC	39522	143082	39650	135957
Cercanos	no-Restr	14688	52141	14688	52266
	no-L-G	15768	55874	15884	55604
Lejanos	no-Restr	38488	160585	38488	160585
	no-K-A-D	41342	171153	44152	167125
16Equipos	no-Restr	63894	262463	64259	253302
	conRestr	63984	254920	63894	254920

Tabla 4.9: Comparación configuración óptima + esquema de visita fijo con parejas dinámicas (PD-EVF)+ 200 iteraciones de la metaheurística (TSPE), contra mejor resultado entre 50 configuraciones de parejas + esquema de visita fijo + 1 iteración de heurística. Todas las configuraciones fueron obtenidas con el MCP.

Esta vez los casos que empeoraban con parejas dinámicas en la Tabla 4.7 mejoraron notablemente. Inclusive una de las instancias sin restricciones obtuvo un mejor fixture sin utilizar la configuración óptima, no obstante, la distancia ganada en el fixture se pierde en la configuración.

Capítulo 5

Conclusiones y trabajo a futuro

En este trabajo hemos propuesto una nueva variante del TTP en base a un caso real, y a su vez, métodos de resolución flexibles y de sencilla aplicación, en vistas de su posible utilización.

Al momento de plantearse, el problema de resolver óptimamente la variación completa (parejas dinámicas y esquema de visita dinámico) resultó intratable. No obstante, atacar los distintos subproblemas por separado, nos ha permitido llegar a soluciones más que aceptables. Podemos afirmar que el problema de obtener un fixture con esquema de visita dinámico, con una configuración de parejas dada, puede resolverse óptimamente (EVD), o aproximadamente con buenos resultados (EVF + TSPE). Por otro lado, a través del uso de distintas soluciones provistas por el MCP dinámico, logramos llegar a buenas soluciones subóptimas para el problema de parejas dinámicas, inclusive hemos demostrado empíricamente que la solución óptima puede no ser de parejas fijas cuando no hay restricciones de localía (instancia A1M-2008-2009 No-Restr).

Desde el punto de vista práctico, al contrario de nuestras expectativas, los experimentos mostraron que tanto la incorporación de parejas dinámicas o esquema de visita dinámico no generan mejoras notables en la distancia recorrida. Incorporar parejas dinámicas sí resulta útil si se da la situación en que las restricciones de localía prohíben aparear dos equipos que debieran mantenerse juntos. El tamaño de instancia con que hemos trabajado (12 equipos) también resulta casi un borde de aplicabilidad de los métodos propuestos, las instancias de 16 equipos ya requieren límites obligatorios de tiempo, lo cual reduce la precisión. Sí destacamos que, en la mayoría de los casos, suele resultar más eficiente hacer una combinación entre un modelo de esquema de visita fijo y la metaheurística, que utilizar modelos que contemplen directamente el esquema de visita dinámico, esto se da especialmente para las ya mencionadas instancias de 16 equipos. También recalamos que la metaheurística no siempre llega al óptimo en la primer iteración, por lo que es recomendable (aparte de no ser costoso), realizar al menos 200 iteraciones.

Al momento de cerrar este trabajo no es posible (de desearse) generar una herramienta totalmente automatizada para obtener buenos fixtures (con

parejas dinámicas y esquema de visita dinámico), es necesario contar con un operador humano que sepa utilizar la batería de procedimientos presentados para buscar un buen fixture a conciencia. De aplicarse, el procedimiento requerirá que el operador pruebe distintas combinaciones de las técnicas propuestas para obtener y refinar soluciones.

Desde el punto de vista teórico, hemos presentado una variante interesante del TTP, junto con una gama de modelos de programación entera que lo representan. Al momento de aplicarlos en el esquema algorítmico, cada modelo ha demostrado superar a los demás en alguna etapa (Modelo 1 en MCP de parejas fijas, Modelo 2 en resolución general y Modelo 3 en la búsqueda local de Intercambio de Rivalidades y MCP de parejas dinámicas). Creemos que en particular el Modelo 2 (modelo en base a grafos) podría llevar a relacionar el TTP con el TSP. Por otro lado, en esta tesis hemos propuesto a modo de innovación intentar realizar búsquedas locales a través de modelos de programación entera, esta idea podría aprovecharse como un paso fuerte en distintas heurísticas y sería interesante estudiarla más a fondo.

Trabajo a futuro

- Encontrar buenas configuraciones de parejas: en nuestro esquema algorítmico, el MCP devuelve configuraciones de parejas bajo un criterio de optimalidad (distancia total recorrida entre Jueves y Sábados) que, si bien consideramos que es correcto, no siempre resulta adecuado. Los últimos experimentos del Capítulo 4 demuestran que esta configuración puede ser decisiva para llegar a una buena solución.
- Probar los modelos con distintas funciones objetivo: nuestros modelos se basan en minimizar la sumatoria de la distancia recorrida entre todos los equipos, pueden probarse variantes como minimizar la distancia recorrida por el equipo que más viaja, o la diferencia entre lo que recorren el que más viaja y el que menos viaja. Estas funciones objetivo pueden ser útiles a la hora de definir un fixture más *justo* según distintos criterios.
- Implementar estos métodos para el sistema de torneo femenino: este torneo cuenta actualmente con 10 equipos (5 parejas fijas) y no tiene weekends interparejas, cada mitad del torneo consta de 5 weekends en los que una sola pareja jugará entre sí mientras las demás se enfrentan normalmente.
- Eliminar la condición de torneo en espejo. Esta propiedad tiene dos finalidades importantes, la primera es ayudar a garantizar un torneo justo en cuanto a distancias viajadas, y en segundo lugar, simplifica enormemente los modelos de programación entera. No obstante, las modificaciones que pueden generar tanto parejas dinámicas como esquema

de visita dinámico en una mitad del torneo son muy susceptibles a tener un correlato muy desfavorable en la otra mitad.

- Estudiar el concepto de plantear las búsquedas locales de las heurísticas como problemas de optimización complejos, no sólo en lo que respecta al TTP.
- Realizar una comparación completa entre el sistema de parejas (algoritmos presentados en este trabajo) y el sistema double round-robin simple (métodos para resolver TTP simple), para analizar si es ventajoso en términos de tiempo y calidad de solución reemplazar el uno por el otro en algún caso.
- Incorporar otras técnicas al esquema algorítmico como pueden ser metaheurísticas comunes, Constraint Programming, o Programación en Paralelo.

Bibliografía

- [1] ACHTERBERG T., *SCIP: Solving Constraint Integer Problems*, <http://scip.zib.de/>
- [2] BONOMO F., BURZYN A., CARDEMIL A. DURÁN G., MARENCO J. (2008) *An application of the traveling tournament problem: The Argentine volleyball league (extended abstract)*. The 7th International Conference on the Practice and Theory of Automated Timetabling - PATAT.
- [3] BURKARD E. R., CELA E., PARDALOS P.M., PITSOULIS L.S. (1984) *The quadratic Assignment Problem*, European Journal of Operational Research 15, 283-289.
- [4] CARDEMIL A. (2002) *Optimización de fixtures deportivos: Estado del arte y un algoritmo tabu search para el traveling tournament problem* (in spanish). MSc Thesis, Universidad de Buenos Aires.
- [5] DE WERRA D. (1980) *Geography, Games and Graphs*, Discrete Applied Mathematics 2 327-337.
- [6] DE WERRA D. (1981) *Scheduling in sports*, in P.Hansen (Ed.), Studies on Graphs and Discrete Programming, North Holland ,381-395.
- [7] DE WERRA D. (1988) *Some models of graphs for scheduling sports competitions*, Discrete Applied Mathematics 21,47-65.
- [8] DI GASPERO L., SCHAERF A. (2007) *A composite-neighborhood tabu search approach to the traveling tournament problem* Journal of Heuristics 13, 2 , 189-207.
- [9] EASTON K., NEMHAUSER G., TRICK M. (2001) *The traveling tournament problem: description and benchmarks*. In Proceedings of the 7th. International Conference on Principles and Practice of Constraint Programming: 580-584.
- [10] EASTON K., NEMHAUSER G., TRICK M. (2003) *Solving the traveling tournament problem: A combined integer programming and constraint programming approach*. Lecture Notes in Computer Science 2740: 100-109.

- [11] GLOVER, F., (1989) *Tabu Search - Part I*, ORSA Journal on Computing 1989 1: 3, 190-206.
- [12] KHACHIYAN L. G., (1979) *A polynomial Algorithm in Linear Programming*, Doklady Akedamii Nauk SSSR, 244, 1093-1096, (English translation: Soviet Mathematics Doklady, 20, 1979, 191-194).
- [13] KOCH T., *Zimpl Modelling Language*, <http://zimpl.zib.de/>
- [14] LACERDA BIAJOLI F., LORENA L. A. N. (2006) *Mirrored Traveling Tournament Problem: An Evolutionary Approach*, Lecture Notes in Artificial Intelligence Series, 208-217.
- [15] LACERDA BIAJOLI F., LORENA L. A. N. (2007) *Clustering Search Approach for the Traveling Tournament Problem*, MICAI 2007: Advances in Artificial Intelligence, 83-93.
- [16] RASMUSSEN R. V., TRICK M. 2006 *Round robin scheduling - A survey*, European Journal of Operational Research, 188, issue 3, 617-636.
- [17] RIBEIRO C.C., URRUTIA S. (2004) *Heuristics for the Mirrored Traveling Tournament Problem*, European Journal of Operational Research, 323-342.
- [18] UTHUS, D. C., RIDDLE P. J., GUESGEN H. W. (2009) *An ant colony optimization approach to the traveling tournament problem*, In Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (Montreal, Québec, Canada). GECCO '09. ACM, New York, NY, 81-88.
- [19] ROSA A. WALLIS W.D. (1992) *Premature sets of 1-factors or how not to schedule round robin tournaments*, Discrete Applied Mathematics 4 ,291-297.
- [20] ROTTEMBOURG B., BENOIST T. (2001) *Lagrange Relaxation and Constraint Programming Collaborative schemes for Traveling Tournament Problems* ,CPAI-OR, WyeCollege, IK , 15-26.
- [21] SCHAERF A. (1999) *Scheduling Sport Tournaments Using Constraint Logic Programming*, Constraints 4 , 43-65.
- [22] TERRIL B.J., WILLIS R.J. (1994) *Scheduling the Australian State Cricket Season Using Simulated Annealing*, Journal of the Operational Research Society 45 / 3, 276-280.
- [23] TRICK M., Challenge Traveling Tournament Instances en <http://mat.gsia.cmu.edu/TOURN/>

- [24] VAN HENTENRYCK P., YANNIS VERGADOS (2006) *Traveling Tournament Scheduling: A Systematic Evaluation of Simulated Annealing*, CPAI-OR : 228-243.
- [25] WOLSEY L. A. (1998) *Integer Programming*, Wiley, New York.
- [26] WUNDERLING R., *SoPlex: The Sequential object-oriented simplex class library*, <http://soplex.zib.de/licence.shtml>

Apéndice A

Instancias de prueba

A continuación presentaremos las instancias de prueba que utilizamos. Las que están basadas en las ligas utilizan matrices de distancias de recorrido reales, las instancias artificiales usan una matriz en base a distancias euclidianas en el mapa.

1. A1M-2007-2008

Primera División de la Liga Masculina, temporada 2007/2008. Ver mapa (A) de la Figura A.1.

a) **no-Restr**

Instancia sin restricciones de localía.

b) **no-FOR-MIS**

Se agrega el siguiente conjunto de restricciones de localía:

- FOR debe jugar de visitante los weekends 4 y 5.
- MIS debe jugar de local los weekends 1 y 2.

Las restricciones de tramo hacen que Formosa y Misiones no puedan ser pareja. Esta situación es interesante porque son dos equipos cercanos muy alejados del resto.

2. A1M-2008-2009

Primera División de la Liga Masculina, temporada 2008/2009. En el caso real fueron once equipos y un equipo artificial situado en la misma ciudad de BOV (Bolívar) en esta instancia hemos agregado el equipo AVC (Azul), cercano a BOV. Ver mapa (B) de la Figura A.1.

a) **no-Restr**

Instancia sin restricciones de localía.

b) **no-CHU-NQN**

Se agrega el siguiente conjunto de restricciones de localía:

- CHU debe jugar de visitante en el weekend 3.
- NQN debe jugar de local en el weekend 3.

Estas restricciones evitan que se pueda formar la pareja más lejana.

c) **no-UPC-OSJ**

Se agrega el siguiente conjunto de restricciones de localía:

- UPC debe jugar de visitante en el weekend 5.
- OSJ debe jugar de local en el weekend 5.

3. A1M-2009-2010

Primera División de la Liga Masculina, temporada 2009/2010. En esta temporada, se decidió evitar que Formosa y Tucumán formaran pareja porque el camino entre las dos ciudades era peligroso para ser recorrido de noche en los lapsos Jueves-Sábado.

Ver mapa (C) de la Figura A.1.

a) **no-Restr**

Instancia sin restricciones de localía. Se permite que Formosa y Tucumán sean pareja.

b) **no-TUC**

Se agrega el siguiente conjunto de restricciones de localía:

- TUC debe jugar de visitante en el weekend 4.
- TUC debe jugar de local en el weekend 5.
- FOR debe jugar de visitante en los weekends 2 y 3.
- UPC debe jugar de local en el weekend 4.
- VMA debe jugar de visitante en el weekend 5.

Esta vez TUC no puede ser pareja fija de FOR a causa de las restricciones de localía, asimismo tampoco puede ser pareja fija de los siguientes dos equipos más cercanos: UPC y VMA. En el caso dinámico queda permitido que Formosa y Tucumán sean pareja si no viola estas restricciones de localía.

4. Cercanos

Instancia artificial con doce equipos cercanos entre sí.

Ver mapa (D) de la Figura A.1.

a) **no-Restr**

Instancia sin restricciones de localía.

b) **no-L-G**

Se agrega el siguiente conjunto de restricciones de localía:

- G debe jugar de visitante en los weekends 2 y 3.
- C debe jugar de local en el weekend 2.
- F debe jugar de visitante en el weekend 3.
- L debe jugar de local en el weekend 2.
- L debe jugar de visitante en el weekend 3.
- A debe jugar de visitante en el weekend 2.
- D debe jugar de visitante en el weekend 3.

Estas restricciones prohíben que los equipos L y G hagan pareja con sus inmediatos más cercanos (A y D, C y F, respectivamente).

5. Lejanos

Instancia artificial con doce equipos lejanos entre sí.

Ver mapa (A) de la Figura A.2.

a) **no-Restr**

Instancia sin restricciones de localía.

b) **no-K-A-D**

Se agrega el siguiente conjunto de restricciones de localía:

- K debe jugar de visitante en el weekend 2.
- K debe jugar de local en el weekend 3.
- D debe jugar de visitante en los weekends 2 y 3.
- A debe jugar de local en el weekend 2.

En este caso, se prohíbe que la terna de equipos más alejada forme parejas fijas.

6. 16equipos

Instancia artificial con 16 equipos. Ver mapa (B) de la Figura A.2.

a) **norestr**

Instancia sin restricciones de localía.

b) **conRestr**

Se agrega el siguiente conjunto de restricciones de localía:

- USH debe jugar de visitante en el weekend 3.
- LPM debe jugar de local en el weekend 4.
- MZA debe jugar de visitante en el weekend 6.
- SNJ debe jugar de visitante en el weekend 5.
- NQN debe jugar de visitante en el weekend 5.

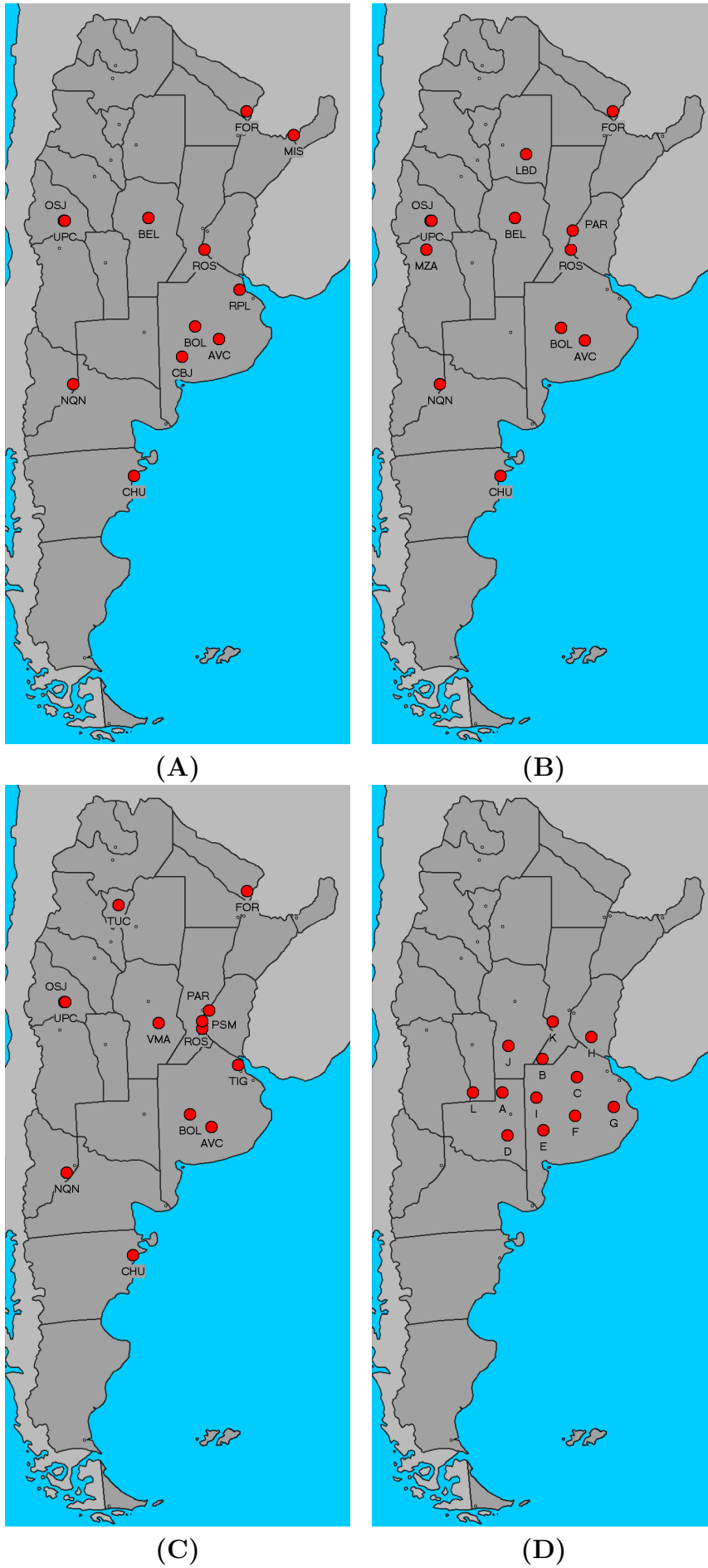


Figura A.1: Mapas: (A): A1M 2007-2008, (B): A1M-2008-2009, (C):A1M-2009-2010, (D):Cercanos

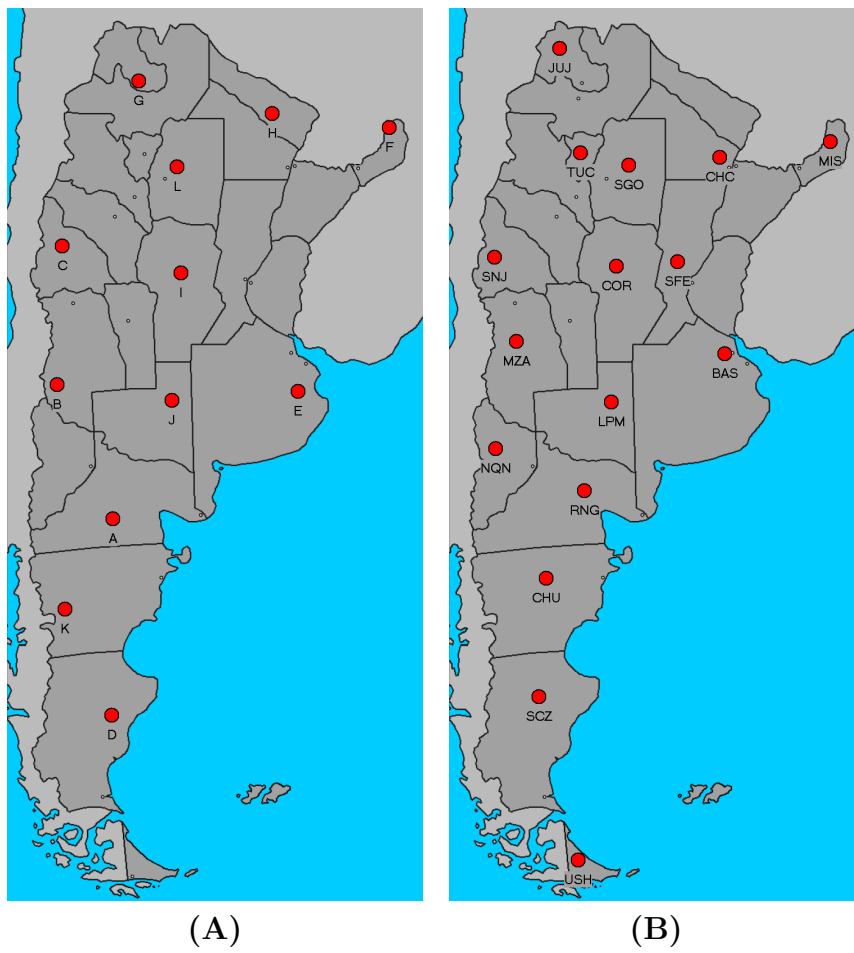


Figura A.2: Mapas: (A): Lejanos, (B): 16Equipos