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

Tesis de Licenciatura

Estado del Arte del Traveling Tournament Problem y una Aplicación a un Caso Real: la Liga Nacional de Básquet de Chile

Florencia Lucila Fernández

Director: Dr. Guillermo Durán

Octubre 2019

Agradecimientos

Quiero agradecer principalmente a mi familia que me apoyó todos estos años y me facilito enormemente el camino para que yo pudiera estudiar esta carrera: mi papa, mi mama, mi hermano y sobre todo mi abuela que me acompañó en cada examen anotando la fecha y deseándome suerte. Sin ellos no hubiera podido llegar hasta acá.

A mis amigos y amigas! A Male, Maru, Mari, Nati y Meli, mis amigas de voley, que siempre me bancaron si no podía juntarme porque tenía que estudiar. Al hermoso grupo de amigos (los Bolzano) que formamos en esta facu. Agus(tina), mi amiga durante toda la carrera, con la cual estudié numerosas veces, pero también encontré a alguien con quien podía compartir también cosas fuera del estudio. A Laion que muchas veces me dio una gran mano explicándome lo que no entendiera, y hasta se juntaba a estudiar conmigo para ayudarme cuando él ya no tenía que estudiar más y siempre me ayudó a calmar nervios antes de los exámenes. Al Gordo, Eicho, Gus, Chanchu (quien nos ha dado clases magistrales de física a todos nosotros en su casa para Temas de Fisica) y Piernis. Todos ellos me apoyaron y ayudaron durante las miles de juntadas de estudio. A Fede B, Vale, Ivo, Nacho, Agus S, Solcito y Colo. A Joaco que me dio una mano enorme (incluso durante fines de semana) con esta tesis. Y a Andi que me ayudó con sus conocimientos de ciencias de la computación y apoyo moral.

Y por último pero no por eso menos importante, agradezco infinitamente a Willy que aceptó ser mi director de tesis y siempre estuvo para guiarme en este último tramo. Además de las oportunidades de viajar, aprender y conocer otros matemáticos que me brindó en el marco de este trabajo de tesis, lo cual fue un broche de oro para cerrar la carrera.

Contents

1 Introducción

El Traveling Tournament Problem (TTP) es un problema de *sports timetabling* que requiere la creación de un torneo double round robin (cada equipo juega contra todos los demás dos veces, una de local y una de visitante) de mínima distancia para un grupo de n equipos.

El TTP no solo es un problema interesante por su forma de modelar las cuestiones de interés real de ligas deportivas, si no que también combina temas de factibilidad relacionados con los patrones local/visitante y temas de optimalidad relacionados con las distancias viajadas. Cabe aclarar que se conjetura que la complejidad computacional del problema es NP-hard. Hay instancias de este problema que parecen ser muy dificiles, aún cuando la cantidad de equipos no es grande, creando un desafío para las técnicas de optimización combinatoria como programación entera y constraint programming (programación con restricciones).

La programación entera es un conjunto de técnicas de la Investigación Operativa dentro de la programación lineal cuando las variables de decisión no pueden tomar valores fraccionarios. Por otro lado, la programación por restricciones es un paradigma de la programación en informática, donde las relaciones entre las variables son expresadas en términos de restricciones (ecuaciones).

1.1 Ventajas

En todo el mundo existen las ligas deportivas profesionales. Más aún, en casos de deportes populares, éstas son de gran importancia económica, ya que la venta de entradas y los derechos de televisación de los partidos ge
neran grandes cantidades de dinero. Por eso es que planificar cuidadosamente el *schedule* de los partidos es extremadamente útil. Un tema importante en la creación de un *schedule* para un torneo es que se especifique el orden en el cual los equipos se enfrentarán durante la temporada y quién será el equipo local. Los equipos y las ligas no quieren desperdiciar sus inversiones en jugadores y logística debido a una mala organización y planificación de los partidos. Crear un *schedule* para un torneo es una tarea difícil que, además de tener muchas restricciones y objetivos, también involucra varias personas

que toman las decisiones. Es decir, que a la hora de llevar a cabo esta tarea, hay que tener en cuenta muchas cuestiones, se trata de reducir costos, pero también, muchas veces, se desea mejorar la competitividad en el torneo, haciendo que el fixture resultante sea más justo, deportivamente hablando.

Usualmente, los equipos desean limitar la cantidad de kilómetros que viajan, pero también les procupan temas más típicos con respecto a los patrones local/visitante. Ningún equipo quiere estar afuera de su ciudad más de dos semanas (lo que corresponde a 3 o 4 partidos, ya que éstos juegan múltiples partidos antes de volver a su ciudad, si es que no tienen ningún partido de local en el medio), ni tampoco se quiere que estén por un período mayor a ése jugando partidos de local, por un tema de equidad deportiva. En algunos casos, como en la liga mayor de baseball (MLB) en Estados Unidos y en ligas de básquet universitario de distintos países, el límite de cantidad de partidos seguidos de visitante se fija en 2.

1.2 Definición Formal

En 2001, Easton, Nemhauser y Trick [7] propusieron una clase de problemas llamada Traveling Tournament Problem, basándose en una serie de trabajos realizados para la liga mayor de baseball (MLB). El problema es especialmente atractivo, entre otras razones, porque, aunque se trate de instancias de relativamente pocos equipos, muchas veces no puede lograrse la optimalidad. Esto lo vuelve interesante y un punto de referencia para otros problemas: es fácil de establecer y el requerimiento de datos es mínimo.

Formalmente hablando, el TTP puede ser presentado como:

Input:

- un conjunto E de equipos donde $|E| = n \geq 4$ es par.
- una matriz de distancias D de $n \times n$. Donde $d_{i,j} \geq 0$ determina la distancia entre los estadios de los equipos y satisface $d_{i,i} = 0$, $d_{i,j}+d_{j,k} \geq d_{i,k}$ (para todo i,j,k) y $d_{i,j} = d_{j,i}$.
- L,U parámetros enteros.

Output: Un torneo *double round robin* de los equipos de E tal que la longitud de cada conjunto de partidos de local o visitante consecutivos sea entre L y U inclusive, y que la distancia total viajada por los equipos se minimice.

Un torneo *double round robin* es aquel en el que los equipos se enfrentan dos veces con todos los demás una vez de local y otra de visitante.

Un partido se representa como un par ordenado de equipos, donde el primero es el local y el segundo el visitante. Una serie de partidos de local (visitante) consecutivos de un equipo se llama *home stand* (*road trip*). También hablaremos de *break* al referirnos al conjunto de dos partidos consecutivos de visitante o de local. Si tenemos n cantidad de equipos, si tenemos un torneo double round robin, se necesitan $2n-2$ fechas. Antes de que comience el torneo, se asume que cada equipo está en su estadio de localía y tiene que regresar allí al finalizar el torneo, en caso de que su último partido sea de visitante. Entre dos partidos de visitante seguidos, un equipo viaja directamente desde el estadio del primer oponente, hasta el estadio del segundo. La longitud de cada *home stand* o *road trip* debe ser entre L y U inclusive. Y, la distancia total viajada debe ser minimizada. Los parámetros L y U definen el tradeoff entre las consideraciones que se tienen de la distancia viajada y el patrón. Por ejemplo, para $L = 1$ y $U = n - 1$, un equipo podría llegar a hacer un *road trip* equivalente al viaje del vendedor ambulante (Traveling Salesman Problem). Para un U pequeño, los equipos deben volver a su casa seguido y, así, la distancia viajada aumentará. Si $U = 1$, la función objetivo se vuelve constante, y el problema sería sólamente de factibilidad. En la práctica, $L = 1$ y $U = 3$ o $U = 2$ son los más usados.

Como dijimos antes, este problema combina temas de factibilidad y optimalidad. Constraint programming es mejor para problemas de factibilidad, y la programación entera, para problemas de optimalidad. Luego, esta mezcla parece ser difícil para ambos métodos, haciendo del TTP un buen problema para explorar la combinación de métodos. Hasta las instancias pequeñas parecen ser difíciles. Mientras que $n = 4$ lleva a instancias fáciles de resolver, $n \geq 6$ ya es un desafío.

La generación de cotas inferiores fuertes es fundamental para lograr optimalidad. Una cota inferior simple se obtiene determinando la distancia mínima a viajar para cada equipo independientemente de cualquier otra re-

stricción que los equipos deseen imponer. La suma de las cotas inferiores de los equipos da una cota inferior que llamaremos *Independent Lower Bound* (ILB) para el TTP. Luego, se puede usar esta cota inferior para abordar el TTP. Para crear una mejor cota inferior (una más alta), primero se busca una buena cota superior.

El concepto de *road trip* es clave. Un *road trip* es una secuencia que describe el viaje que realiza un equipo. Cada elemento de la secuencia está asociado con un intervalo de tiempo en el *schedule* y proporciona también, la localía para ese partido. Por ejemplo, en una instancia con $n = 6$, un conjunto de *road trips* para un equipo 1 podría ser $(1, 2, 3, 4, 1, 1, 5, 1, 6, 1)$. Notar que el estadio de cada oponente aparece exactamente una vez y el estadio del equipo en cuestion aparece $n - 1$ veces. O sea que dicho equipo juega una vez de visitante contra cada uno de los restantes equipos y todas las demás veces de local contra todos los restantes equipos (exactamente una vez con cada uno).

La clave es ordenar las soluciones por la cantidad de *road trips* que realizan los equipos. En general, menos *road trips* significan menos distancia viajada porque significa que los equipos no tienen que regresar a su casa tan seguido. Un patrón es un vector de designaciones de local/visitante, uno por cada lugar del vector. Y un conjunto de patrones es una colección de patrones, una por cada equipo. Es fácil generar conjuntos de patrones en orden creciente de la cantidad de *road trips*. Para un conjunto de patrones dado, es un problema mucho más fácil forzar una solución para que matchee, es decir, que coincida con ese conjunto, y es la base de una gran parte de la literatura de sports scheduling. Entonces, podemos generar conjuntos de patrones incrementando la cantidad de *road trips* y encontrar la distancia mínima para cada conjunto. Una vez que se tiene una solución factible, se puede agregar una restricción que solo quiera mejores soluciones, lo que luego hará que el tiempo de cómputo sea más rápido.

De todas maneras, no queremos trabajar con todos los conjuntos de patrones: hay demasiados hasta para n=6. En vez de hacer eso, se puede modificar la ILB para incluir la mínima cantidad de restricciones posibles para los *road trips*. Una vez que la ILB con esta restricción está por encima de nuestra solución factible, sabemos que no necesitamos considerar ningún patrón con más *road trips*.

Este método, generalmente, encuentra muy buenas soluciones rápidamente y puede probar optimalidad para instancias chicas. Para instancias más grandes, se ha trabajado con una combinación de métodos de programación entera y constraint programming que incluyen la generación de columnas. En estos modelos, las variables corresponden a estructuras de un nivel más alto, incluyendo *road trips*, homestands, e incluso *schedule* s completos. Los métodos de constraint programming se usan para generar variables que luego se combinan con técnicas de programación entera. El éxito de ello depende ampliamente del conjunto inicial de variables y de las reglas de branching (ramificación) que se usan.

1.3 Instancias Importantes

Se han propuesto dos clases de problemas para los experimentos algorítmicos del TTP. La primera es un conjunto artificial de instancias designadas para determinar el efecto de los aspectos del problema del vendedor ambulante en el TTP. Y la segunda es una serie de instancias de la MLB (liga mayor de baseball), que proveen la inspiración inicial para el estudio del TTP en sus comienzos.

La primera clase está conformada por las *Instancias Circulares*. Los argumentos de la complejidad del TTP se relacionan con el problema del vendedor ambulante (TSP). De todas maneras, no queda claro que el TTP sea fácil incluso cuando el TSP es trivial. Cuando se explora esto con la instancia donde el TSP es fácilmente resoluble (y para la cual la solución es única), el TTP sigue siendo desafiante para resolver.

La instancia circular de n nodos tiene distancias generadas por el grafo circular de n nodos con distancias unitarias. En este grafo, los nodos son etiquetados como $1, ..., n-1$; hay una arista del i a $i+1$ y de $n-1$ al 0, cada una con longitud 1. La distancia de i a j (con $i > j$) es la longitud del menor camino en este grafo, y es igual al mínimo entre $i-j$ y $j-i-n$. En este grafo, $0, 1, ..., n-1$ da el tour de TSP óptimo.

La segunda clase está conformada por las *Instancias de la Liga Nacional de Baseball*. Desafortunadamente, la MBL tiene demasiados equipos para

el actual estado del arte de encontrar soluciones óptimas. Se divide en dos ligas: la nacional y la americana. Casi todos los partidos que cada equipo juega son en contra de equipos de su propia liga, así que es razonable limitar el análisis a una liga sola. Se han generado las matrices de distancias de la liga nacional usando la distancia aérea entre los centros de las ciudades. Para generar instancias más chicas, simplemente se toman subconjuntos de los equipos. Haciendo esto, se crearon las instancias NL4, NL6, NL8, NL10, NL12, NL14 y NL16, donde el número dado es la cantidad de equipos en esa instancia.

1.4 El Single Team Problem

Se propone al TTP como un problema de referencia por las siguientes razones:

1. Tiene una importancia práctica, ya que colabora con el modelado de problemas de sports scheduling en la vida real.

2. La mezcla entre factibilidad y optimalidad, hace al problema muy interesante para la investigación operativa y para constraint programming.

Easton et al. [8] dieron resultados para algunas formulaciones básicas. Además, presentaron un enfoque que combina métodos de programación entera y *constraint programming* para resolver problemas más grandes mucho más rápido.

Para entender mejor el análisis, es conveniente mirar el *Single Team Problem* (STP), donde se evalúa el problema desde la perspectiva de un equipo. El *road trip* óptimo de un equipo minimiza la distancia viajada por ese equipo, más allá de los demás equipos en el *schedule* .

Para la mayoría de los valores de L y U, generar simplemente un *road trip* óptimo para un equipo determinado es NP-hard, a pesar de que existan algunos casos especiales resolubles en tiempo polinomial. Se realizó un estudio de la complejidad de generar un *road trip* de mínima distancia para cada

combinación (factible) de L y U. Se asume que la desigualdad triangular se mantiene para las distancias.

Formalmente, se define el Single Team Problem de la siguiente forma:

Input: un conjunto de n equipos $T = \{t_1, ..., t_n\}$; un equipo designado $t_r \in T$; una matriz entera de distancias D simétrica de n por n con elementos $d_{i,j}$ que satisfacen la desigualdad triangular; L, U parámetros enteros, un valor límite entero k.

Output: Existe un *road trip* para t_r tal que

- la longitud de cada *home stand* y cada *road trip* tenga longitud entre L y U inclusive, y
- la suma de las distancias viajadas por t_r es menor o igual a k?

Para L y U específicos, usamos la notación $STP(L, U)$.

En los extremos de L y U, la complejidad de $STP(L, U)$ es clara. Para $STP(1, 1)$, la función objetivo es constante y la solución se puede obtener alternando los partidos de visitante y local. Para $STP(L, N-1)$, el problema es un Traveling Salesman Problem y es NP-completo. De hecho, mientras U sea proporcional a n^α para una constante fija α, hay una reducción inmediata a partir del TSP para probar NP-completitud.

El STP también es NP-completo para toda constante $U > 2$ y $L = U$. La reducción es a partir del problema NP-completo *Partition into Isomorphic Subgraphs Restricted to Paths* (PISP).

Más adelante hablaremos de la complejidad del TTP cuando no se tienen restricciones para L y U, y veremos que es NP-hard [2].

En el caso especial en que $U = 2$ y $L = 1$ (es la única asignación factible para L, pues n es par), un algoritmo $O(n^3)$ puede ser obtenido si se usan matchings ponderados para encontrar el *road trip* óptimo. Para eso, primero notar que siempre va a haber un *road trip* óptimo con cantidad mínima de viajes. Dos viajes de un equipo cada uno, pueden ser transformados en un sólo viaje de dos equipos con una

distancia no mayor a la de la suma de los dos viajes de equipos individuales por la desigualdad triangular. Segundo, como n es par, el conjunto mínimo de viajes va a incluir siempre un viaje de exactamente un equipo.

Dicho esto, para encontrar el *road trip* de distancia mínima, Easton et al. [9]se tiene el siguiente algoritmo:

Algoritmo STP(1,2):

1. Dado un equipo t_r, se genera un grafo $G = (V, E)$ completo con n nodos. Sea el vértice i representante del equipo t_i para todo $i \in \{1, ..., n\}$. Asignar pesos a las aristas de esta forma: sea $w_{pq} = d_{rp} + d_{pq} + d_{qr}$ para todo $p, q \in \{1, ..., n\} \setminus \{r\}$, $p \neq q$. Se fija $w_{rp} = 2 * d_{rp}$ para todo $p \in \{1, ..., n\} \setminus \{r\}$
2. Encontrar un matching perfecto de peso mínimo en G.
3. Crear un *road trip* de distancia mínima a partir del matching: para cada par matcheado de nodos (p, q) tal que $p, q \in \{1, ..., n\} \setminus \{r\}$, crear un *road trip* que sea de dos partidos con t_r jugando en t_p y luego en t_q. Para el nodo p que está matcheado con r, crear un *road trip* de un solo equipo en el cual t_r juega en t_p. Además, crear un conjunto de breaks de local que imiten los viajes. Es decir, un break de local con un solo intervalo de tiempo y $\frac{n}{2} - 1$ breaks de local con dos viajes. Finalmente, crear el *schedule* del *road trip* eligiendo alternadamente breaks de visitante y de local y poniéndolos consecutivamente en los intervalos de tiempo del *schedule* del torneo.

Teorema 1: El algoritmo STP(1,2) devuelve una solución óptima.

Demostración: Todo *road trip* factible con una cantidad mínima de viajes corresponde a un matching perfecto de G con la distancia del *road trip* igual al peso del matching. Luego, encontrar un matching perfecto de peso mínimo en G devuelve una solución óptima para el STP(1,2).

El tiempo de corrida de este algoritmo esta dominado por el trabajo que conlleva resolver el matching. Un matching perfecto ponderado en un grafo genérico puede ser encontrado en $(|E||V| + log(|V|)|V|^2) = O(n^3)$. Luego, el algoritmo corre en $O(n^3)$. Este algoritmo puede ser fácilmente generalizado en el caso en que la desigualdad triangular no se mantenga, usando b-matchings en vez de matchings perfectos (un b-matching de un grafo dado G es una asignación de pesos enteros a

las aristas de G tal que la suma de los pesos de las aristas incidentes en un vértice v es a lo sumo b_v, donde b denota el vector de b_v's. Cuando $b_v = 1$ para todo vértice de G, entonces el b-matching es un matching usual.

1.4.1 Una metodología de solución mediante Branch and Price

Easton et al. presentaron una metodología de solución para el TTP, usando el algoritmo branch-and-price (generación de columnas), en el cual, viajes de equipos individuales son las columnas. En branch-and-price, la relajación de programación lineal en el nodo raíz del árbol de branch and bound incluye sólo un pequeño subconjunto de las columnas. Para chequear la función objetivo de la programación lineal, se resuelve un subproblema llamado *pricing problem*, para determinar si hay columnas adicionales disponibles para entrar a la base. Si el pricing problem devuelve una o más columnas, la programación lineal se reoptimiza. Si no se pueden encontrar más columnas para entrar a la base y la solución de la programación lineal es fraccionaria, el algoritmo se ramifica. Branch-and-price es una generalización del branch and bound con relajaciones de programación lineal. En este enfoque combinado, se usa contraint programming para resolver el pricing problem.

Más allá de que la cantidad de equipos en las instancias del TTP que se han podido resolver parecen pequeñas, estos problemas son, de hecho, bastante grandes en términos de las cantidades de columnas posibles. Para una instancia con 8 equipos hay 4.5 millones de *road trips*. Para poder resolver estas instancias en un tiempo razonable, se corre una version paralela del algoritmo de branch-and-price. Se usa la dialéctica del amo y el esclavo. Luego de generar el conjunto inicial de nodos, el procesador amo pasa un nodo a cada esclavo junto con la mejor solución conocida hasta el momento (si es que hay). Cada esclavo evalúa un número fijo de nodos y pasa de vuelta su mejor solución y todos los nodos que ha generado, especificando qué nodos han sido evaluados y cuáles no. Si un esclavo descifra por completo su porción del árbol, no le devuelve ningún nodo al amo. El amo mantiene la mejor solución reportada por los esclavos y una lista de todos los nodos no evaluados. Un nodo de esta lista y la mejor solución hasta el momento son pasados al ahora esclavo inactivo para ser procesados. Cuando la lista del amo está vacía y todos los procesadores están desocupados, el algoritmo está completo y la mejor solución conocida hasta el momento es óptima.

Se comienza en el nodo raíz con un pequeño conjunto de columnas $P = \{1, ..., m\}$ compuesto por los *road trips* óptimos para cada equipo. Sea el vec-

tor c que representa las distancias asociadas con los *road trips*; T el conjunto de equipos del torneo y S el conjunto de intervalos de tiempo. Sea P_t los índices de los *road trips* para el equipo $t \in T$ y x una variable de decisión binaria que representa los *road trips*. Esto nos lleva a un programa entero:

Minimizar $\sum_{i \in P} c_i x_i$

sujeto a

$\sum_{i \in P_t} x_i = 1 \qquad \forall t \in T$

$\sum_{\{i \notin P_t \text{ y t es oponente en el intervalo de tiempo s en i}\}} x_i +$

$\sum_{\{i \in P_t : t \text{ juega de visitante en el intervalo de tiempo s en i}\}} x_i = 1 \qquad \forall s \in S, \forall t \in T$

x_i binaria para todo $i \in P$

El primer conjunto de restricciones fuerza a que se elija exactamente un *road trip* para cada equipo en el torneo. El segundo conjunto de restricciones le pide a cada equipo que juegue exactamente una vez por intervalo de tiempo.

En algunos algoritmos de branch-and-price, las columnas son generadas hasta que no quedan más columnas de costo reducido negativo. En otros, las columnas son generadas sólo si un nodo está por ser cortado. La rutina de branch and price descripta recién pertenecería a la segunda categoría de algoritmos. Una de las razones por las cuales pasa eso es que si se generan menos columnas, el problema amo resolverá más rápido. Lo que es más importante, sin embargo, es que el problema de pricing es más fácil de resolver en el segundo caso. Así que, generar columnas sólo cuando sea necesario, reduce el tiempo que toma resolver ambos problemas: el del amo y el de pricing. Por otro lado, no se puede usar una estrategia de selección del mejor nodo cota si no siempre genero todas las columnas. Con *parallel programming*, es difícil de llegar a mejor cota en cualquier momento dado. Esta técnica usa múltiples recursos, en este caso procesadores, para resolver un problema particular. Este tipo de programación toma un problema, lo divide en una serie de pasos de menor tamaño y procesadores diferentes ejecutan sus soluciones al mismo tiempo. También es una forma de programación que ofrece resultados en menos tiempo que

otras y con mayor eficiencia.

Investigaciones previas que usan algoritmos de branch-and-price indican que es más eficiente generar una reserva de columnas, elegir de ésta en cada nodo, luego generar una nueva reserva cuando sea necesario. Así, en cada nodo de nuestro arbol de búsqueda, primero chequeamos si hay alguna columna de costo reducido negativo en la reserva. De ser así, elegimos hasta 10 columnas por cada equipo. Si no, rellenamos la reserva. El problema de pricing se resuelve usando constraint programming.

Cuando se invoca, el contraint program para el subproblema es corrido una vez por cada equipo. Genera todos *road trips* de costo reducido negativo. Las variables son los estadios en los cuales los partidos de los equipos se juegan en cada intervalo de tiempo. Los dominos de estas variables son los equipos del torneo. Dos variables binarias se agregan para los intervalos de tiempo 0 y $2(n-1)+1$. Estas variables se establecen igual al equipo local y son usadas para calcular distancias. Las distancias entre ciudades consecutivas también son parámetros pero son usadas estrictamente para calcular la función objetivo. Antes de que se corra el constraint program, los dominios de las variables son reducidos de acuerdo al branching previo.

Las restricciones de este modelo son las siguientes (para L=1, U=3):

- El estadio de cada oponente aparece exactamente una vez.
- No pueden aparecer más de 3 partidos de local consecutivos.
- No pueden aparecer más de 3 partidos de visitante consecutivos.
- El estadio local aparece exactamente $n-1$ veces.
- La suma de las distancias y los valores duales asociados con los partidos del *road trip* deben ser negativos (equivalentemente, el costo reducido del *road trip* resultante debe ser negativa).

Las variables se eligen en orden de tamaño del dominio. En el caso en que todas las variables no instanciadas tienen dominios del mismo tamaño, la variable que esté más cerca al final del torneo (ya sea el primer o el último intervalo de tiempo no instanciado) es elegida. Los empates en este segundo criterio son desempatados arbitrariamente. Una vez que una variable es elegida, será instanciada con el equipo en su dominio que haga la contribución más negativa (o menos positiva) al costo total del *road trip* que incluye las distancias entre partidos consecutivos y los

costos reducidos asociados con oponentes e intervalos de tiempo.

La instanciación de una variable, resulta en la reducción del dominio para las restantes variables no instanciadas. Si un partido de visitante es asignado, el algoritmo primero elimina ese oponente de futuras consideraciones. Si no fuera así, un partido de local estaría siendo asignado y el estadio local estaría siendo eliminado de futuras consideraciones sólo si el número del partido de local alcanza $n - 1$. Además, se consideran las restricciones de patrones. Cualquier elemento de los dominios restantes que pueda crear breaks de visitante o de local mayores a 3 intervalos de tiempo es eliminado. Finalmente, se considera la contribución hacia el objetivo de asignaciones potenciales. La generación de nuevas columnas es un procedimiento simple, y no se ha observado mejoras en el tiempo de corrida luego de agregar un cierto grado de visión hacia al futuro a este algoritmo. Si un dominio es reducido a ningún elemento restante, se realiza un backtracking al último punto de decisión. Si un dominio es reducido a uno, se marca la variable. Cuando se completa una iteración de constraint propagation, el algoritmo elige una variable marcada (si es que hay), la instancia con el único miembro de su dominio y otra iteración de constraint propagation.

Se encontró que es útil poblar la reserva con "buenos" *road trips* (distancias relativamente pequeñas) además de aquellos con costos reducidos negativos. Luego, se agregan *road trips* a la reserva con distancias dentro de un incremento de la Cota Inferior Independiente, aumentando este incremento iterativamente. Esto reduce la cantidad de veces que necesitamos rellenar la reserva.

Como se discutió anteriormente, *parallel programming* hace que determinar la mejor solución en cualquier momento sea más difícil. Por eso, se ha usado una estrategia de búsqueda en profundidad (DFS).

Más que ramificar en un *road trip*, se usa variables de ramificación de altos órdenes. Las variables de altos órdenes que se usaron son los patrones indexados por equipo e intervalo de tiempo. En otras palabras, en cualquier nodo dado, en el cual el valor del programa lineal amo sea menor que la mejor solución hasta ese momento, se divide el espacio de solución entre *schedule* s en los cuales el equipo t es local en el intervalo de tiempo s y *schedule* s en los cuales el equipo t es visitante en el intervalo s. Con el objetivo de elegir una variable de orden más alto en la cual ramificarse, se usa una estrategia conocida como *strong branching*. Se crea una lista de candidatos seleccionando las 10 variables de orden más alto con peso total más cercano a 0.5. Para cada variable en la lista de candidatos, se realizan 50 iteraciones del método simplex, seteando la variable igual a 0 en un caso e igual

a 1 en el segundo caso. Luego, se elige la variable con el mayor cambio total en el valor de la función objetivo sumado sobre los dos casos.

La componente final de esta metodología de solución es una heurística que hace uso de una versión expandida del constraint program del pricing problem. Esta heurística trabaja en el *schedule* completo y el modelo se modifica de manera acorde. Específicamente, los arreglos de variables son expandidos para incluir n equipos, y se agregan las siguientes restricciones:

- No más de dos equipos pueden jugar en un mismo estadio en un mismo intervalo de tiempo.

- Si un equipo juega de visitante, su oponente debe jugar de local.

El objetivo es minimizar las distancias sumadas de todos los equipos e intervalos de tiempo.

Sólamente con correr este modelo no obtenemos "buenas" soluciones lo suficientemente rápido; sin embargo, si fijamos ciertos elementos se puede generar soluciones con el 5-6% de la cota inferior independiente en una cantidad de tiempo razonable. Específicamente, se eligen $\frac{n}{2}$ equipos, cada uno de los cuales es forzado a jugar uno de sus *road trips* óptimos. Notar que esto no significa que elegimos $\frac{n}{2}$ columnas para fijar. El conjunto de *road trips* óptimos de un equipo incluye aquellos con distancia mínima. "Fijar" un equipo consta de sumar un conjunto de restricciones al modelo que se relacionen con los viajes que el equipo seleccionado podría jugar. Una vez que se eligió un conjunto de equipos fijos, se corre el constraint program para un intervalo de tiempo específico exponiendo soluciones y soluciones mejoradas a medida que van siendo generadas.

Al final del intervalo de tiempo seteado, se elige un nuevo conjunto de cuatro equipos y se corre el modelo de nuevo. Los equipos fijados son elegidos en orden de proximidad al centro estimado de la región geográfica definida por los equipos que participan en el torneo. Un procesador se dedica a correr la heurística. El procesador amo chequea el archivo output de la heurística en intervalos que dependen del tamaño de la instancia. Si se encuentra una mejor solución entera, ésta es enviada a procesadores esclavos junto con los nodos siguientes que se le han dado para procesar.

1.5 Instancias del Problema

Como se mencionó anteriormente, el TTP fue creado para capturar la esencia de ligas deportivas reales, en particular de la Liga Mayor de Baseball (MLB). Desafortunadamente, la MLB tiene muchos equipos, 30 exactamente, para el estado-del-arte actual del TTP para encontrar soluciones óptimas. La MLB está dividida en dos ligas: la Liga Nacional y la Liga Americana. En casi todos los partidos, los equipos juegan contra equipos de su propia liga, con lo cual es razonable limitar el análisis a una liga individual, para tener así, menos equipos para analizar.

Para generar instancias aún menores, se toman subconjuntos de los equipos. Haciendo esto, se crean las instancias NL4, NL6, NL8, NL10, NL12, NL14 y NL16, en las cuales el número indica la cantidad de equipos en esa instancia. Todas esas instancias están incluidas en una *challenge page*, de la cual hablaremos en detalle más adelante, asociada a este trabajo: http://mat.gsia.cmu.edu/TOURN. Allí se suele imponer la condición en la que dos equipos no podrían enfrentarse dos partidos seguidos dos fechas consecutivas. Y eso debe ser considerado a la hora de hacer comparaciones con otros trabajos de investigación.

Instancias con $n = 4$ se resuelven casi de manera trivial. Instancias con $n = 6$ son más desafiantes. Easton et al. han encontrado varios modelos que pueden resolver estas instancias en una cantidad de tiempo razonable sin *parallel programming*. Cuando usaron 20 procesadores para resolver instancias con $n = 6$, el tiempo de corrida era del orden de minutos. Finalmente, encontraron que es necesario usar *parallel programming* para resolver instancias con $n = 8$ equipos.

La tabla a continuación muestra tiempo reloj para poder encontrar y probar soluciones óptimas para NL4, NL6 y NL8 en ese momento, al resolverlos con una red de PCs con procesadores 300MHz Pentium II y RAM de 512MB corriendo Redhat Linux 7.1.

Nombre	L	U	ILB*	Solución Óptima	Procesadores	Tiempo (s)
NL4	1	3	8276	8276	1	30
NL6	1	3	22969	23552	20	912
NL8	1	3	38670	39479	20	362630

*ILB: Cota Inferior Independiente - medida en millas

Además, obtuvieron las cotas para NL16, corriendo el constraint program de la heurística que se mencionó anteriormente en un sólo procesador por 24 horas de

corrida. Se obtuvo una cota inferior de 248852 y una cota superior de 312623.

1.6 Challenge Page

Como esta clase de problemas se creó en el contexto del Major League Baseball, la página web lleva un registro de las diferentes instancias que se resolvieron y sus mejores soluciones factibles y cotas inferiores hasta el momento.

Las reglas que se imponen para estas intancias del TTP son:

- Double Round Robin (A juega en B y B juega en A): n equipos necesitan 2*n-2 intervalos de tiempo (*slots*).
- No puede haber más de tres partidos de local o visitante consecutivos para ningún equipo (cuando se especifica que la entrada registrada es "unconstrained", implica que esta restricción no se considera).
- No se puede repetir el mismo partido dos veces seguidas (no puede suceder A en B, seguido inmediatamente de B en A).
- El objetivo es minimizar la distancia viajada (asumiendo que los equipos empiezan en su ciudad local y deben retornar allí al finalizar el torneo).
- Los nombres de las ciudades están en el siguiente orden: ATL NYM PHI MON FLA PIT CIN CHI STL MIL HOU COL SF SD LA ARI. Donde NLx toma las primeras x ciudades.

Ésta es el registro hasta el momento (donde "@" indica localía):

➢ **NL4 (4 teams):**
Solución Factible: 8276
Cota Inferior: 8276

ATL	NYM	PHI	MON
PHI	MON	@ATL	@NYM
NYM	@ATL	MON	@PHI
MON	@PHI	NYM	@ATL
@PHI	@MON	ATL	NYM
@NYM	ATL	@MON	PHI
@MON	PHI	@NYM	ATL

➢ **NL6 (6 teams):**
(Trick May 1, 1999)
Solución Factible: 23978
Cota Inferior: 22969

Slot	ATL	NYM	PHI	MON	FLA	PIT
0	FLA	@PIT	@MON	PHI	@ATL	NYM
1	NYM	@ATL	FLA	@PIT	@PHI	MON
2	PIT	@FLA	MON	@PHI	NYM	@ATL
3	@PHI	MON	ATL	@NYM	PIT	@FLA
4	@MON	FLA	@PIT	ATL	@NYM	PHI
5	@PIT	@PHI	NYM	FLA	@MON	ATL
6	PHI	@MON	@ATL	NYM	@PIT	FLA
7	MON	PIT	@FLA	@ATL	PHI	@NYM
8	@NYM	ATL	PIT	@FLA	MON	@PHI
9	@FLA	PHI	@NYM	PIT	ATL	@MON

(Trick May 1, 1999)
Solución Factible: 3916
Cota Inferior: 23916

ATL	NYM	PHI	MON	FLA	PIT
@FLA	@PHI	NYM	PIT	ATL	@MON
@PHI	PIT	ATL	@FLA	MON	@NYM
MON	@FLA	PIT	@ATL	NYM	@PHI
NYM	@ATL	@MON	PHI	@PIT	FLA
@PIT	PHI	@NYM	FLA	@MON	ATL
@MON	@PIT	FLA	ATL	@PHI	NYM
@NYM	ATL	MON	@PHI	PIT	@FLA
PIT	MON	@FLA	@NYM	PHI	@ATL
PHI	FLA	@ATL	@PIT	@NYM	MON
FLA	@MON	@PIT	NYM	@ATL	PHI

➢ **NL8 (8 teams):**

Solución Factible: 39721 *(Easton, August 25 2002)*
Cota Inferior: 39479 *(Easton Feb 26, 2002)*

	ATL	NYM	PHI	MON	FLA	PIT	CIN	CHI
1	FLA	@PIT	@CIN	CHI	@ATL	NYM	PHI	@MON
2	PIT	CHI	@FLA	CIN	PHI	@ATL	@MON	@NYM
3	MON	CIN	CHI	@ATL	PIT	@FLA	@NYM	@PHI
4	@PIT	@CHI	CIN	@FLA	MON	ATL	@PHI	NYM
5	@CHI	@CIN	MON	@PHI	@PIT	FLA	NYM	ATL
6	@CIN	PHI	@NYM	PIT	@CHI	@MON	ATL	FLA
7	CHI	PIT	@MON	PHI	@CIN	@NYM	FLA	@ATL
8	CIN	MON	PIT	@NYM	CHI	@PHI	@ATL	@FLA
9	@MON	@PHI	NYM	ATL	CIN	CHI	@FLA	@PIT
10	@NYM	ATL	@PIT	FLA	@MON	PHI	@CHI	CIN
11	@PHI	FLA	ATL	@CHI	@NYM	@CIN	PIT	MON
12	NYM	@ATL	FLA	@CIN	@PHI	@CHI	MON	PIT
13	PHI	@FLA	@ATL	@PIT	NYM	MON	CHI	@CIN
14	@FLA	@MON	@CHI	NYM	ATL	CIN	@PIT	PHI

Solución Factible: 41505 *(Easton June 4, 1999)*, 41113 *(Easton Jan 27, 2000)*, 40416 *(Cardemil, July 2 2002)*, 39947 *(Zhang, August 19 2002)*
Cota Inferior: 38760 *(Trick Dec 15, 2000)*, 38870 *(Easton Jan 2, 2001)*, 39721 *(Irnich and Schrempp, June 24 2008)*

➢ **NL10 (10 teams):**
Solución Factible: 68691 *(Rottembourg and Laburthe June 2001)*, 66369 *(Dorrepaal, June 21 2002)*, 66037 *(Cardemil, July 2 2002)*, 64597 *(Zhang August 6, 2002)*, 61608 *(Zhang August 19, 2002)*, 59583 *(Van Hentenryck January 14, 2003)*, 59436 *(Langford, June 13, 2005 solution)*.
Cota Inferior: 56506 *(Waalewijn July 2001)*, 57500 *(Easton June 2002)*, 57817 *(Irnich and Schrempp, June 24 2008)*, 58831 *(Uthus, Riddle, and Guesgen, Feb 11 2009)*, 59436 *(Uthus, Riddle, and Guesgen December 11, 2009)*.

➢ **NL12 (12 teams):**
Solución Factible: 143655 *(Rottembourg and Laburthe May 2001)*, 125803 *(Cardemil, July 2 2002)*, 119990 *(Dorrepaal July 16, 2002)*, 119012 *(Zhang, August 19 2002)*, 118955 *(Cardemil, November 1 2002)*, 114153 *(Anagnostopoulos, Michel, Van Hentenryck and Vergados January 14, 2003)*, 113090 *(Anagnostopoulos, Michel, Van Hentenryck and Vergados February 26, 2003)*, 112800 *(Anagnostopoulos, Michel, Van Hentenryck and Vergados June 26, 2003)*, 112684 *(Langford February 16, 2004)*, 112549 *(Langford February 27, 2004)*, 112298 *(Langford March 12, 2004)*, 111248 *(Anagnostopoulos, Michel, Van Hentenryck and Vergados May 13, 2004)*, 110729 *(Van Hentenryck and Vergados, May 30 2007)*.
Cota Inferior: 107483 *(Waalewign August 2001)*, 107494 *(Melo, Ribeiro, and Urrutia July 15 2006)*, 107548 *(Mitchell, Trick and Waterer July 31 2008)*, 108244 *(Uthus, Riddle, and Guesgen, Feb 11 2009)*, 108629 *(Uthus, Riddle, and Guesgen January 6, 2010)*.

➢ **NL14 (14 teams):**
Solución Factible: 301113 *(Rottembourg and Laburthe June 2001)*, 262010 *(Larichi, Lapierre, Laport July 8 2002)*, 216108 *(Cardemil, July 2 2002)*, 207075 *(Zhang, August 28 2002)*, 205894 *(Cardemil, November 20 2002)*, 195555 *(Anagnostopoulos, Michel, Van Hentenryck and Vergados January 14, 2003)*, 190368 *(Van Hentenryck February 26, 2003)*, 190056 *(Langford April 22, 2004)*, 189766 *(Anagnostopoulos, Michel, Van Hentenryck and Vergados May 13, 2004)*, 189759 *(Dorrepaal and Chackman, April 13, 2005)*, 188728 *(Van Hentenryck and Vergados, May 18 2006)*.
Cota Inferior: 182797 *(Waalewign August 2001)*, 183354 *(Uthus, Riddle, and Guesgen January 29 2010)*.

➢ **NL16 (16 teams):**
Solución Factible: 312623 *(Easton January 2002)*, 308413 *(Cardemil, July 2 2002)*, 301256 *(Zhang August 6 2002)*, 293175 *(Zhang, August 28 2002)*, 284235 *(Shen, October 16 2002)*, 281660 *(Shen, January 6 2003)*, 277766 *(Anagnostopoulos, Michel, Van Hentenryck and Vergados January 14, 2003)*, 273802 *(Anagnostopoulos, Michel, Van Hentenryck and Vergados February 26 2003)*, 267194 *(Van Hentenryck, June 26, 2003)*, 263772 *(Van Hentenryck and Vergados, May 18 2006)*, 261687 *(Van Hentenryck and Vergados, May 30 2007)*.
Cota Inferior: 248852 *(Easton January 2002)*, 249477 *(Melo, Ribeiro, and Urrutia July 15 2006)*.

2 Sports Scheduling

El *Sports Scheduling* es un área dentro de la programación matemática que se encarga del armado de un fixture en una competencia deportiva. Para eso, debe tener en cuenta ciertas condiciones que van a depender del tipo de deporte, del tipo de torneo, del requerimiento de la asociación correspondiente, de los requerimientos de los equipos que participan en dicho torneo, etc. Se podría decir que se divide en dos categorías: problemas de *time relaxed* scheduling (temporalmente relajado) y de *time constrained* scheduling (temporalmente restringido).

En el caso del time constrained, la cantidad de partidos es igual al mínimo requerido de espacios de tiempo y para el time relaxed, la cantidad de espacios de tiempo disponibles es más grande que el minimo requerido.

En los ultimos 30 años, se realizó mucha investigación en el tema, la cual mayormente se enfocó en *time constrained scheduling*. Los objetivos de la programación pueden clasificarse en dos grupos: la minimización de breaks y la minimización de la distancia viajada. Cuando se juegan partidos consecutivos de visitante o de local, puede no ser equitativo, deportivamente hablando, para todos los equipos, o algunos equipos se vean afectados por cansancio, mayores costos, etc. Las ligas quieren un *schedule* con la mínima cantidad de breaks posibles, ya que, si los equipos viajan largas distancias, esto podría causar que los jugadores y el entrenador se sientan fatigados, lo cual terminaría siendo una ventaja para los oponentes.

Si nos remontamos a la historia de la construcción de *schedules* para torneos deportivos profesionales y amateur, se han utilizado varias herramientas matemáticas, por ejemplo la 1-factorización. En 1980, la relación entre 1-factorizaciones de grafos y los torneos, fue explotada por de Werra [21]. Notar que un factor de un grafo es un subgrafo del grafo G que tiene el mismo conjunto de vértices que G. Luego, un k-factor de un grafo G es un subgrafo de G k-regular. Con lo cual, una k-factorización particiona las aristas del grafo G en k-factores disjuntos. Un torneo SRR compacto, se puede asociar con un grafo completo. Cada nodo corresponde a un equipo y cada arista corresponde a un partido entre los equipos asociados a los dos nodos finales. Para construir una 1-factorizació de un grafo completo, se utilizan n-1 1-factores correspondientes a una partición de los partidos en n-1 rondas. La orientación de la arista puede representar la localía de ese partido.

Gracias a la teoría de grafos, los investigadores pudieron desarrollar teorías muy importantes. Por ejemplo, la cota inferior de breaks para un *schedule* SRR es $n-2$. Pues, si un equipo no tiene ningún break, tiene que alternar el patrón de localía

cada vez. Esto se debe a que, como cada par de equipos tiene que enfrentarse al menos una vez, no puede haber dos equipos con el mismo patrón. Solamente hay dos patrones de localía posibles sin breaks: empezar siendo local o empezar siendo visitante. Como resultado de esto, como mucho, dos equipos pueden tener patrón de localía sin break. Todo el resto de los n-2 equipos tiene al menos una ronda con la localía diferente a la de los patrones mencionados, lo cual genera un break. Originalmente, el método de usar grafos se usaba para resolver problemas sin ninguna otra restricción. A pesar de que de Werra [22] y otros investigadores desarrollaron algunas soluciones intrincadas para resolver el problema con restricciones, el uso del método del grafo es limitado si hay muchas restricciones.

Se realizaron estudios sobre métodos de descomposición, porque es realmente efectivo para abordar problemas complicados. Un problema de sports scheduling se puede descomponer en cuatro pasos: generar los patrones, generar los conjuntos de patrones, generar el esquema de partidos y generar el *schedule* completo. Trick (2001) argumentó que el orden debería ser arreglado en base a las dificultades de estos pasos. Para obtener un *schedule* con la cantidad de breaks mínima, uno querría determinar primero los patrones y los conjuntos de patrones. Esto nos hace cuestionarnos la factibilidad de los conjuntos de patrones. Si un *schedule* opuesto se programa primero, se podría utilizar programación entera y constrained programming para resolver el problema de minimización de breaks.

En los 1990s, se comenzó a implementar varios métodos metaheurísticos para minimizar breaks en problemas de la práctica. Willis y Terrill [24] usaron recocido simulado y Wright [25] uso búsqueda tabú para el *schedule* de un torneo de cricket.

La mayoría de las ligas profesionales deportivas usan time constrained *schedules*, por ejemplo, el básquet en universidades de Estados Unidos, las ligas profesionales de fútbol de Europa, etc. Pero hay dos ligas deportivas profesionales en Norteamérica que usan time relaxed *schedules*: la liga nacional de hockey (NHL) y la liga nacional de básquet (NBA). Durante los ultimos 30 años se trató de resolver este tipo de problemas.

Uno de los pocos estudios enfocados en el problema de scheduling de la NBA, fue realizado por Bean y Birge [1], cuyo objetivo fue reducir la distancia viajada. Dicen que el problema no se puede resolver con programación lineal porque la cantidad de variables es muy grande. Así es que proponen una heurística de dos fases. La primera, es crear los *road trip* para cada equipo; y la segunda, combinar éstos de acuerdo a las restricciones. Los *road trip* pueden ir de 1 a 5 partidos. El más largo se programa para el período en el que haya menor disponibilidad en el

estadio de dicho equipo. Un *road trip* se puede dividir en *road trips* parciales en caso de que no pueda programarse en ningún lugar del *schedule*. Los autores de esta investigación reportaron un resultado del 20% de ahorro de distancia viajada comparado con el *schedule* oficial.

Hasta el momento hubo varias investigaciones enfocadas en el scheduling de la NHL. Fraser [11] (1982) desarrolló un modelo de un "simulador de *road trips*". Declaró que es imposible usar la computadora para generar el *schedule* sin ajustarlo manualmente de alguna forma. Por otro lado, Ferland y Fleurent [10] (1991) propuso un sistema de soporte de toma de decisiones que ayudaba a los usuarios a crear interactivamente el *schedule*. Hay dos componentes importantes en esta herramienta: la programación de los *road trips* y las herramientas de intercambio. La primera parte es programar los *road trips* requeridas para todos los equipos. Los *road trips* fueron clasificadas como *forzadas* y *libres*. Si tenemos un caso en el que no hay disponibilidad en algún estadio por un período largo, esto causará un *road trip* forzado. Se usó una heurística similar a la que usaron Bean y Birge para construir estos *road trips* forzados. Este método tiene cuatro etapas. La primera es seleccionar el equipo que tenga un período de no disponibilidad de su estadio más largo. La segunda, programa los dos partidos con mayor índice de distancia minimizada en la mitad de la temporada. Los partidos candidatos a esto son los que no pueden ser disputados en días consecutivos debido a la distancia entre sus estadios. La tercera es programar los partidos del final del *road trip* con la misma lógica usada en la etapa anterior. La última etapa consiste en programar los partidos del comienzo del *road trip*. El procedimiento de intercambio se trata de dos opciones: intecambio simple o intercambio doble. En el primero, el partido que queda puede ser programado cambiando un partido. Similarmente, en el intercambio doble, el partido que queda se puede programar cambiando dos partidos. Este sistema produjo muy buenos resultados.

Costa [6] (1995) también investigó sobre el tema y propuso un algoritmo evolutivo de búsqueda tabú. Todas las restricciones son clasificadas en dos tipos: *esenciales* y *relajadas*. Todos los *schedules* factibles satisfacen las restricciones esenciales. Las restricciones relajadas se usan para juzgar la calidad de la solución. Hay cuatro pasos en este algoritmo. El primero, es generar soluciones iniciales. El método usado en el primer paso es similar al usado por Ferland y Fleurent [10]. El segundo paso es reproducir las soluciones. Cuanto más apta sea la solución, más chance va a tener de ser elegida para la reproducción. El tercer paso, es realizar un *entrecruzamiento* (crossover) de soluciones. Se trata de poner tantos partidos como sea posible en una solución sin cambiar el marco. Todos los partidos que sean redundantes, se quitan de acuerdo a los costos. En el paso final, se aplica la

búsqueda tabú. En cada iteración, los partidos que violan al menos una restricción relajada, se eligen para cambiar el día del partido. El último movimiento, se considera como una búsqueda tabú para las siguientes iteraciones. Solamente una parte de las restricciones se usa para construir la función de aspiración. Este método tiene buenos resultados.

Un caso particular es el de la Liga alemana de Básquet (BBL). Westphal[23] (quien, recordemos, figura en la *Challenge Page*) discute el problema de encontrar un *schedule* óptimo para la temporada 2011-2012. Una de las mayores dificultades enfrentadas, fue que la mayoría de los partidos se llevan a cabo en estadios de múltiples usos que también son utilizados para otros eventos, con lo cual no siempre están disponibles. Además, se debe minimizar el tamaño de los breaks, asignar los partidos más "interesantes" para las televisoras a los intervalos de tiempo designados para televisar, minimizar la distancia que los equipos deben viajar durante todo el torneo y tratar de satisfacer los requerimientos sobre las localías por parte de los equipos.

Se presentan varios algoritmos y se muestra cómo estos modelos cumplen los requerimientos de la BBL. En pos de eso, se muestra que los modelos clásicos, los cuales eran aplicados anteriormente (y que todavía son aplicados por otras ligas), son limitados a la hora de cumplir dichos requerimientos. También, se muestra que los *schedules* canónicos no poseen las propiedades que se desearían y que los *schedules* espejados no cubren las necesidadesd de la liga, convenciéndolos de utilizar *schedules* no espejados por primera vez en la historia.

La BBL consta de 18 equipos que juegan en contra en 34 rondas. Cada equipo juega un partido por ronda durante toda la temporada y se enfrentan en un double round robin. Los partidos se organizan tal que los dos partidos que cada equipo juega contra otro, se den en distintas mitades de la temporada y cada equipo juega al menos 8 y como mucho 9 partidos de local en cada mitad.

Además, es mejor jugar un partido un fin de semana, ya que eso implica que los fans tienen más posibilidad de ir a ver el partido, lo que implica más ingresos mediante las entradas vendidas. Por eso, las rondas se distribuyen durante la temporada de manera tal que los 30 fines de semanas de la temporada, correspondan a una ronda. Eso nos dejaría 4 rondas restantes, que se programan para 4 miércoles diferentes en el medio de la temporada.

Como mencionamos anteriormente, la mayoría de los partidos se disputan en localidades que tienen diferentes usos, con lo cual, no se dispone de algunos estadios

en algunas fechas. Con la estrategia que se utilizaba anteriormente para determinar el fixture, la mayoría de los partidos debían ser reprogramados, lo cual generaba muchos inconvenientes económicos y con respecto a la organización. Además, como el partido terminaba siendo reprogramado para un día en el medio de la semana, se vendía una cantidad notablemente menor de entradas para el evento deportivo. Ni que hablar de la situación en la que varios partidos de un mismo equipo eran reprogramados para la misma semana. A esto, se le suma el inconveniente de que, llegado el fin de semana, no todos los equipos habían jugado la misma cantidad de partidos, con lo cual, no se podía hacer una comparación acertiva de las performances deportivas de los equipos. En realidad, en ningún punto del torneo, sucedía que todos los equipos tuvieran la misma cantidad de partidos jugados. Todos estos problemas a la hora de programar un torneo son críticos para la liga alemana, por lo que significa un aspecto fundamental el tener en cuenta estas restricciones de disponibildad de cancha.

Más aún, se podría asumir que un simpatizante deportivo promedio no desea ir a un evento deportivo cada fin de semana, aunque esa persona tampoco querrá tener que esperar mucho tiempo para poder asistir al siguiente partido. Luego, se podría decir que aún el mayor fanático no desearía viajar todos los fines de semana a diferentes localidades. Es por eso que suena lógico querer diseñar el fixture de manera tal que a cada partido de local le siga un partido de visitante y que a cada partido de visitante le siga uno de local. Es decir que se trata de evitar los breaks.

No debemos olvidar que la transmisión televisiva es un punto importante. Las fechas de transmisión televisiva están disponibles para 30 rondas. Es por eso que los partidos que suponen la mayor cantidad de televidentes, deberían ser programados para estas fechas. Estos partidos se denominan *partidos-A*, y los restantes partidos se denominan *partidos-B*. Entonces, los *schedules* de la BBL incluyen 20 partidos-A y 60 partidos-B.

La BBL asume que los simpatizantes no van a viajar en año nuevo, con lo cual, se desarrolló el concepto de *derby* (una ronda en la cual solo programan partidos entre equipos que tengan localía cerca entre sí. Para poder cumplir eso, se trató de minimizar la distancia total entre los equipos que se enfrentan entre sí ese día.

Por último, algunos equipos tienen ciertas preferencias en cuanto al *schedule* de partidos de local o visitante en días específicos. Por ejemplo, el estadio podría estar sometido a reformas o puede ser utilizado para otros eventos no necesariamente deportivos en ciertas fechas, etc; éstas son preferencias de localía.

Además de las ligas profesionales anteriormente mencionadas como la NBA y la NHL, algunos torneos profesionales también usan time relaxed scheduling. Uno de ellos es el torneo amateur de tenis de mesa en Alemania. Fue un problema de scheduling de torneo DRR. Las restricciones que se consideraron en esta investigación incluían días libres, balance de la cantidad de partidos entre equipos y disponibilidad de cancha. Schonberger [17] (2004) modeló este problema como un problema de constraint programming. Las variables son partidos entre los equipos, los valores las fechas posibles para cada partido. Los autores intentaron usar contrained programming, pero no obtuvo buenos resultados cuando la cantidad de equipos se volvía grande. Por eso, se propuso un algoritmo genético para resolver el problema. La variable usada en el modelo de Constraint Satisfaction Problem se elige según el código del algoritmo genético. El operador de mutación es intercambiar el dia de partido de dos partidos. Para llevar la búsqueda hacia la factibilidad, la función objetivo penalizaba la violación de las restricciones. Además, se implementó una heurística de mejora local. Y, aquí también, se obtuvieron buenos resultados.

3 Complejidad del Problema

El TTP tiene muchas preguntas abiertas, incluso la de la complejidad. Bhattacharyya [2] mostró que, si no se considera la restricción que limita la longitud de los breaks de local o de visitante, el TTP es NP-hard.

Se mostró, entonces, que si los equipos tienen permitido jugar cualquier cantidad de partidos seguidos de local o visitante, luego, hay una reducción a partir de una instancia del (1,2)-Traveling Salesman Problem, es decir, encontrar un *road trip* TSP en un grafo donde el costo de cada arista es 1 o 2.

Veamos la demostración. Primero, debemos observar que si sólo un equipo tuviera que viajar, luego, la solución óptima del TTP hubiera sido el Traveling Salesman Tour de las ciudades. Pero cuando más de un equipo debe movilizarse, no parece conveniente tratar de analizar la sincronización de los viajes de cada uno de esos equipos de esa manera. Para resolver el TSP usando un oráculo para resolver el TTP, se puede construir una instancia del TTP con 2n-2 equipos. Se colocan n-1 equipos en una ciudad c (llamados equipos "buenos") y un equipo en cada una de las restantes n-1 ciudades. Ahora, todos los equipos buenos pueden viajar fuera de c en las mismas rondas, y jugar sus partidos de visitante sin volver a casa. Notar que un *schedule* así va a ser factible cuando el parámetro U sea n (la cantidad de ciudades). Luego, se debería fijar el parámetro de tal manera cuando se muestre una reducción. Esto implica que el costo de viaje de cada equipo de c es exactamente el Traveling Salesman tour del grafo. Por otro lado, cuando se acota el peso de cada arista, podemos acotar el costo de viaje de los demás equipos. A pesar de que esta cota puede no ser muy exigente, esta diferencia se va a volver muy pequeña comparada con el costo total cuando la cantidad de equipos en c aumente.

Para acotar por abajo al costo total de viaje del torneo, observemos que el costo de viaje de cualquier equipo debe ser al menos el costo del TSP tour de las ciudades. Luego, se construye un grafo a partir de la instancia dada que amplifique el costo del Traveling Salesman tour de la instancia original. En otras palabras, si la instancia que tenemos como input tiene un costo K del TSP tour, el grafo construido tendrá un costo de TSP tour de nK. Entonces, si el costo del TSP tour óptimo de la instancia se vuelve K+1, el costo del tour óptimo del grafo construido se volverá nK+n. Ahora, si se puede diseñar una instancia con un *schedule* double round robin donde el costo de viaje de cada equipo es menor que nK+n, podremos reducir el problema de TSP a un TTP. Rercordar que K es el costo del TSP tour en la instancia que tenemos como input.

En cuanto a la notación, debemos tener en cuenta que:

- $[t] = 1, 2, ..., t$; el conjunto de los primeros t números naturales.

- $G = (V, E)$ es un grafo completo pesado sin aristas paralelas o loops. V es el conjunto de vértices y E es el conjunto de aristas.

Se utiliza el input del TTP clásico. Y se realiza la siguiente **pregunta**: ¿existe un *schedule* de n equipos localizados en las ciudades representadas en la matriz de distancias D tal que la cantidad de partidos consecutivos de local o visitante esté entre L y U, con distancia total viajada por todos los equipos a lo sumo K?

Cada instancia de este problema se representa TTP(n,D,L,U,K). Sin pérdida de generalidad, podemos asumir que D es una matriz es una métrica, ya que cualquier matriz de distancia en la vida real será una métrica. Para mostrar que el TTP es NP-hard, se muestra una reducción a partir de una variante del Traveling Salesman Problem; llamado (1,2)-Traveling Salesman Problem ((1,2)-TSP). Luego, el problema de decisión de (1,2)-TSP se define así:

→ **Problema:** (1,2)-Traveling Salesman Problem

→ Un set de enteros $C = c_1, ..., c_n$ de n ciudades, Distancia $d(c_i, c_j) \in 1, 2$ para todo par $c_i, c_j \in C$, Un entero K

→ **Pregunta:** ¿existe un Traveling Salesman tour de C con un costo de a lo sumo K? En otras palabras, ¿existe una permutación $\pi : [n] \to [n]$ tal que $\sum_{i=1}^{n-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(n)}, c_{\pi(1)}) \leq K$?

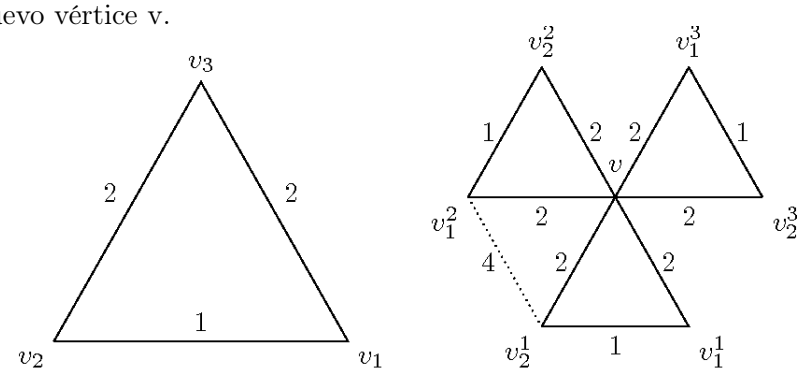
Es fácil corroborar que (1,2)-TSP es NP-hard. Uno puede construir una reducción a partir del problema del *Ciclo Hamiltoniano* como sigue: sea $G = (V, E)$ una instancia del problema del problema del Ciclo-Hamiltoneano con $|V| = n$. Uno puede construir una instancia (C, d, K) del (1,2)-TSP haciendo $C = V$ y $K = n$. Finalmente, $d(v_i, v_j)$ es 1 si $(v_i, v_j) \in E$ y 2 si no. Ahora G tiene un ciclo Hamiltoniano en G si y sólo si es un tour de C con costo a lo sumo n.

Para la construcción, tendremos en cuenta ciertas definiciones. Sea (n, D, K) una instancia de (1,2)-TSP. Denotamos las n ciudades como $c_1, ..., c_n$. Veremos

la instancia como un grafo ponderado completo $G = (V, E)$ con $|V|$ y $d(v_i, v_j) = D(c_i, c_j)$ para todo $i, j \in [n]$ donde $d : E \to 1, 2$ es la función de costo de las aristas. Nuestro objetivo es construir un grafo G^1 con costo nK. Tal grafo puede ser fácilmente construido de la siguiente forma:

Fijamos un vértice $v \in V$. Consideremos n copias distintas de G y contraemos todas las copias de v en un sólo vértice. Así, el nuevo grafo tiene $n^2 - n + 1$ vértices. Llamamos v al vértice central de G^1. El costo de las aristas se calcula así: si dos vértices u, w pertenecen a la misma copia de G, luego $d'(u, w) = d(u, w)$; si no, $d'(u, w) = d(u, v) + d(v, w)$.

A continuación, mostramos un ejemplo de la construcción de G^1 con tres copias de G, si G tuviera tres vértices, e hicieramos la contracción del vértice v_3 en un nuevo vértice v.



(a) The input instance (b) Corresponding G^1

Es fácil ver que tenemos el siguiente hecho:

Lema 1: G^1 tiene un traveling salesman tour de costo nK si y sólo si D tiene un traveling salesman tour de costo K.

Ahora, estamos en condiciones de realizar la reducción formal de (1,2)-TSP a TTP. Sea (n, d, K) la instancia del input del (1,2)-TSP. Se construye el grafo G^1 a partir de esa instancia como se describió anteriormente. Recordar que G^1 tiene $m = n^2 - n + 1$ vértices. Sean los vértices de G^1 denotados como $v_1, ..., v_m$. Sin pérdida de generalidad, asumimos que v_1 es el vértice central de G^1. Construimos un grafo H a partir de G^1 agregando otro vértice u a G^1. Cada arista (u, v_i) tiene peso $1000(m + 1)^4$.

Luego, se construye la instancia TTP correspondiente con $n' = 10(m+1)^2$ equipos. Colocamos $m+1$ equipos en G^1, de los cuales dos equipos están en el vértice (central) v_1 y $(m-1)$ equipos están en los vértices restantes (cada uno de los vértices de G^1 distintos de v_1, tiene exactamente un equipo). Colocamos el resto de los $10(m+1)^2 - (m+1) = (m+1)(10m+9)$ equipos en el vértice u. Sea D' la matriz de distancias del grafo H. Se fija $L = 1$ y $U = n'$, lo cual implica que no hay, efectivamente, restricción para la cantidad de partidos consecutivos de local o visitante. En el siguiente lema, se muestra una cota superior del costo total de viaje de un *schedule* óptimo del TTP-(n',D',1,n').

Lema 2: Si una instancia (n, D) con $n > 5$ tiene un traveling salesman tour de costo K, luego el TTP(n',D',1, n') tiene un *schedule* factible de costo a lo sumo $20000(m+1)^6 + 10m(m+1)(nk+1)$ donde $m' = n(n-1)+1$, $n' = 10(m+1)^2$ y D' es construido como se mostró anteriormente.

Demostración: Sea τ un Traveling Salesman tour en (n, D), la instancia de input de TSP, con costo K. Por el lema 1, G^1 tiene un traveling salesman tour τ' de costo nK. Se podría, entonces, construir un *schedule* factible de la instancia TTP(n', D', 1, n') con costo total de viaje a lo sumo $20000(m+1)^6 + 10m(m+1)(nk+1)$. Primero, se dividen los $10(m+1)^2$ equipos en $10(m+1)$ grupos, cada uno de tamaño $m+1$. Todos los eequipos de G^1 están en el mismo equipo, lo llamamos Grupo 1. Para el resto de los equipos (equipos en el vértice u), los grupos se forman arbitrariamente.

Un *schedule* de un torneo double round robin de n' equipos consiste en un total de $2(n'-1)$ rondas. Un equipo juega 2m partidos (m de local y m de visitante) contra equipos de su propio grupo y el resto de los $2(m+1)(10m+9)$ partidos, juega contra equipos de otros grupos. En nuestro *schedule*, en las primeras 2m rondas, los equipos van a jugar todos los partidos contra los equipos de su propio grupo. Y el resto de los partidos serán jugados entre grupos.

Primero, se necesita un *schedule* factible de un torneo double round robin de $m+1$ equipos. El siguiente lema, muestra que dicho *schedule* es posible.

Lema 3: Existe un *schedule* single round robin de n equipos para todo n par.

En este caso, la cantidad de equipos en cada grupo es $m+1 = n(n-1)+2$, que es un número par. Más aún, $L = 1$ y $U = n' > n$, asegura que cualquier *schedule* double round robin es factible. Ahora, se puede construir fácilmente un *schedule* double round robin repitiendo el *schedule* del lema 3, con el patrón local-visitante

invertido.

El costo total de viaje de estas 2m rondas puede ser acotado de la siguiente manera. Todos los partidos de los equipos que no sean del Grupo 1, están en el vértice u. Luego, los equipos en el vértice u no necesitan viajar en esas rondas. Entonces, no hay costo de viaje para ellos. Por la desigualdad triangular en el grafo G^1, el costo de viaje de cada equipo del Grupo 1, es a lo sumo dos veces el peso de las aristas incidentes en el vértice correspondiente. Recordar que G^1 tiene un total de m vértices y dos equipos están en u y cada uno de los otros vértices tienen un equipo. Luego, el costo total de viaje para el equipo en el vértice v_i del Grupo 1 es $\leq 2\sum_{j=1,j\neq i}^{m+1} w(v_i, v_j) \leq 8m$. Entonces, el costo total incurrido en las primeras 2m rondas es a lo sumo $8m(m+1)$.

El resto de los partidos serán jugados dentro de los grupos. Luego, debe verse el resto del *schedule* como un torneo *dummy* de $10(m+1)$ equipos donde el equipo i representa al grupo i. Un partido de dicho torneo, disputado entre el grupo i y el grupo j, es, de hecho, m+1 rondas de los partidos donde cada equipo del grupo i juega contra cada miembro del grupo j. Más aún, si el grupo i juega los partidos de local, entonces, todos los miembros del grupo i jugarán los partidos de local en estas m rondas.

Primero consideremos cualquier torneo single round de $10(m+1)$ equipos. En todos los partidos de estos $10m+9$ rondas, asignamos partidos de visitante para el equipo 1. A partir del argumento anterior, esto implica $(10m+9)(m+1)$ partidos de visitante para cada equipo del Grupo 1 en el vértice u. Como todos los demás equipos están en el vértice u, todos estos partidos de estas $(10m+9)(m+1)$ rondas son jugadas en el vértice u.

Recordar que al final de las primeras 2m rondas $(m+1)$, los equipos del Grupo 1 estaban en algunos vértices (no necesariamente en su localía) en G^1 y todos los demás equipos estaban en u. En las próximas $(10m+9)(m+1)$ rondas, sólo los equipos del Grupo 1 viajan al vértice u. Como el costo de una arista desde cualquier vértice de G^1 al vértice u es $1000(m+1)^4$, el costo total de viaje de estas rondas es $1000(m+1)^4 \times (m+1) = 1000(m+1)^5$.

Para las últimas $(10m+9)(m+1)$ rondas, se repite el mismo *schedule* entre los grupos, con el patron local-visitante invertido. Luego, todos los equipos del Grupo 1 juegan partidos de local en estas rondas. Todos los demás grupos van a G^1 y juegan sus partidos de visitante. Recordar que cada grupo juega exactamente m partidos en G^1 en m rondas consecutivas. Como hay m vértices en G^1,

los equipos que juegan de visitante necesitan viajar. Para ilustrar este patrón de viaje, supongamos que el grupo i visita al grupo 1. Primero, cada equipo juega con su correspondiente equipo; i.e. el equipo 1 del grupo i juega de visitante en la ciudad del equipo 1 del grupo 1, el equipo 2 del grupo i juega de visitante en la ciudad del equipo 2 del grupo 1, etc. En la siguiente ronda, los equipos del grupo i siguen lo que indica τ'.

Para calcular el costo de estas $(10m+9)(m+1)$ rondas, notar que los equipos del grupo 1 necesitan volver a sus correspondientes vértices (su ciudad local) en G^1. Este viaje tiene un costo de $1000(m+1)^4 \times (m+1) = 1000(m+1)^5$. Luego, cada grupo va a G^1 y sigue acorde al traveling salesman tour. Observar que cada equipo viaja exactamente por m aristas en G^1. Luego, un costo total de viaje para el grupo i es $\leq 2 * 1000(m+1)^5 + (m+1)nK - nK = 2000(m+1)^5 + mnK$.

Como $2(10m+9)(m+1)+2m = 2*(10m^2+20m+10-1) = 2(10(m+1)^2-1)$, estas rondas terminan el *schedule*. Entonces, el costo total de viaje del *schedule* es como mucho $(10m+9)(2000(m+1)^5 + mnK) + 2000(m+1)^5 + 8m(m+1) = 20000(m+1)^6 + (10m+9)mnK + 8m(m+1) < 20000(m+1)^6 + 10m(m+1)nK + 8m(m+1) < 20000(m+1)^6 + 10m(m+1)(nK+1)$. ■

Lema 4: Si la instancia input (n, D) con $n > 5$ no tiene un traveling salesman tour de costo menor o igual que K, luego TTP(n', D', 1, n') no tiene *schedule* factible de costo menor o igual a $20000(m+1)^6 + 10m(m+1)(nK+1)$.

Demostración: Para cualquier *schedule*, el costo de viaje de cada equipo es a lo sumo el costo del traveling salesman tour de las ciudades. Suponer que el TSP tour óptimo de la instancia input tiene costo $K+k$ donde $k \geq 1$. Por lema 1, el traveling salesman tour óptimo de G^1 tiene costo $n(k+k)$. Recordar que cada arista de G^1 tiene costo a lo sumo 4. El costo de las aristas que unen u con cualquier vértice de G^1 es $2000(m+1)^4$. Luego, el traveling salesman tour óptimo de H tiene costo a lo sumo $200(m+1)^4 + n(K+k) - 4$. Luego, el costo del torneo será al menos $10(m+1)^2(2000(m+1)^5 + n(K+k) - 4) = 20000(m+1 =^6 +10(m+1)^2nK + 10(m+1)^2(nk-4)$. Ahora para cada $n > 5$, el costo total de viaje del torneo es mayor a $20000(m+1)^6 + 10m(m+1)(nK+1)$. ■

Usando el lema 4 y el 2, llegamos al siguiente resultado:

Teorema: El Traveling Tournament Problem sin restricciones en cuanto a la cantidad de partidos consecutivos de local o visitante que pueden jugar los equipos es NP-hard.

4 Variantes del TTP

El TTP tiene diversas variantes de su formulación clásica, entre las cuales podemos mencionar:

- ▷ No Round Robin: los equipos no juegan double round robin, sino que hay valor de "matchup" (pareja) entre los equipos i y j, que indica la cantidad de veces que i debe visitar a j. El TTP clásico es un Non-RR TTP con un valor de matchup de 1 para todo i distinto de j.

- ▷ Sin Repetidores: dos equipos no pueden jugar dos partidos seguidos entre sí (aunque cambie la localía).

- ▷ Con Repetidores: dos equipos podrían jugar dos partidos seguidos entre sí.

- ▷ Espejado: es un DRR TTP donde los partidos jugados en los intervalos de tiempo s son los mismos que se juegan en los intervalos de tiempo $s+(n-1)$ para todo $s = 1, ..., n-1$. Claramente, aquí queda implícita la restricción de "sin repetidores".

- ▷ Time Relaxed: la cantidad de intervalos de tiempo (slots) disponibles es más grande que el mínimo requerido.

- ▷ Time Constrained: la cantidad de partidos es igual al mínimo requerido de intervalos de tiempo.

- ▷ Multi Round Balanced: inspirado en la estructura de la liga japonesa de béisbol profesional, donde n=6 equipos juegan 120 partidos dentro de su liga durante r=8 rondas sujeto a varias restricciones que aseguran balance en la competitividad.

4.1 Instancias de variantes de TTP

- ◦ Campeonato de Fútbol de Brasil: aquí se consideran ciudades de Brasil, en el torneo del 2003 con 24 equipos (n=24).

Solución Factible: 506433 *(Urrutia and Ribeiro August 26, 2004) (mirrored)*, 503158 *(Urritia, Ribeiro and Araujo, March 11 2005) (mirrored)*, 500,756 *(Van Hentenryck and Vergados, May 30 2007) (mirrored).*

- Super Liga de Rugby 14: compuesta por equipos de Nueva Zelanda, Australia y Sudáfrica.

  - Super4 (4 equipos):
    Solución Factible: 63405 *(Uthus, Riddle, and Guesgen, Feb 11 2009)*
    Cota Inferior: 63405 *(Uthus, Riddle, and Guesgen, Feb 11 2009)*

  - Super6 (6 equipos):
    Solución Factible: 130365 *(Uthus, Riddle, and Guesgen, Feb 11 2009)*
    Cota Inferior: 130365 *(Uthus, Riddle, and Guesgen, Feb 11 2009)*

  - Super8 (8 equipos):
    Solución Factible: 182409 *(Uthus, Riddle, and Guesgen, Feb 11 2009)*
    Cota Inferior: 182409 *(Uthus, Riddle, and Guesgen, Feb 11 2009)*

  - Super10 (10 equipos):
    Solución Factible: 316329 *(Uthus, Riddle, and Guesgen, Feb 11 2009)*
    Cota Inferior: 316329 *(Uthus, Riddle, and Guesgen, Feb 11 2009)*

  - Super12 (12 equipos):
    Solución Factible: 467267 *(Misir, Wauters, Verbeeck, Vanden Berghe, March 30 2009)*, 465667 *(Uthus, Riddle and Guesgen April 29 2009)*, 463876 *(Langford May 8, 2009)*, 460998 *(Shirai and Miyashiro, Jan 29 2016)*
    Cota Inferior: 367812 *(Uthus, Riddle, and Guesgen, Mar 11 2009)*, 452597 *(Uthus, Riddle, and Guesgen June 28 2009)*, 453860 *(Uthus, Riddle, and Guesgen January 6, 2010)*

  - Super14 (14 equipos):
    Solución Factible: 599296 *(Misir, Wauters, Verbeeck, Vanden Berghe, March 30 2009)*, 586435 *(Uthus, Riddle and Guesgen April 29 2009)*, 573669 *(Langford May 4, 2009)*, 571632 *(Langford May 15, 2009)*
    Cota Inferior: 467839 *(Uthus, Riddle, and Guesgen, Mar 11 2009)*,

557070 *(Uthus, Riddle, and Guesgen June 28 2009)*, 557354 *(Uthus, Riddle, and Guesgen January 29 2010)*

- Liga de Football Nacional (NFL): basada en equipos de la liga nacional de Estados Unidos de fútbol americano, provee intancias grandes para experimentación. Las instancias más grandes, son hasta 32 equipos.

  - NFL16 (16 equipos):
    Solución Factible: 241973 *(Langford, Dec 30 2005)*, 241264 *(Di Gaspero and Schaerf, January 27, 2006)*, 241214 *(Langford, March 13 2006)*, 238581 *(Di Gaspero and Schaerf, March 27, 2006)*, 237428 *(Langford, April 28, 2006)*, 235930 *(Langford, October 10, 2006)*, 231,483 *(Van Hentenryck and Vergados, May 30 2007).*
    Cota Inferior: 223800 *(Melo, Ribeiro, and Urrutia July 15 2006).*

  - NFL18 (18 equipos):
    Solución Factible: 301546 *(Di Gaspero and Schaerf, Jan 09 2006)*, 299192 *(Langford, Jan 10 2006)*, 298705 *(Di Gaspero and Schaerf, March 3 2006)*, 296638 *(Langford, April 18 2006)*, 282,258 *(Van Hentenryck and Vergados, May 30 2007).*
    Cota Inferior: 272834 *(Melo, Ribeiro, and Urrutia July 15 2006)*

  - NFL20 (20 equipos):
    Solución Factible: 361878 *(Di Gaspero and Schaerf, Jan 09 2006)*, 353927 *(Langford, Jan 10 2006)*, 352947 *(Di Gaspero and Schaerf, Feb 16 2006)*, 350256 *(Langford, April 24, 2006)*, 346324 *(Langford, April 25, 2006)*, 332041 *(Van Hentenryck and Vergados, May 30 2007).*
    Cota Inferior: 316721 *(Melo, Ribeiro, and Urrutia July 15 2006).*

  - NFL22 (22 equipos):
    Solución Factible: 439626 *(Di Gaspero and Schaerf, Jan 09 2006)*, 426977 *(Langford, Jan 18, 2006)*, 418,086 *(Urrutia and Ribeiro, January 22 2006)*, 412812 *(Langford, May 12, 2006)*, 402534 *(Van Hentenryck and Vergados, May 30 2007)*
    Cota Inferior: 378813 *(Melo, Ribeiro, and Urrutia July 15 2006).*

  - NFL24 (24 equipos):
    Solución Factible: 499017 *(Di Gaspero and Schaerf, Jan 09 2006)*,

490,528 *(Langford, Jan 20, 2006)*, 467,135 *(Urrutia and Ribeiro, January 22 2006)*, 463,657 *(Van Hentenryck and Vergados, May 30 2007)*
Cota Inferior: 431,226 *(Trick, June 22, 2007)*

- 🏆 NFL26 (26 equipos):
  Solución Factible: 590,497 *(Di Gaspero and Schaerf, January 27 2006)*, 573,596 *(Langford, January 27, 2006)*, 554,670 *(Urrutia and Ribeiro, February 10 2006)*, 551,033 *(Langford, June 29 2006)*, 536,792 *(Van Hentenryck and Vergados, May 30 2007)*.
  Cota Inferior: 495,982 *(Trick, June 22, 2007)*

- 🏆 NFL28 (28 teams):
  Solución Factible: 681,197 *(Di Gaspero and Schaerf, January 27, 2006)*, 669,320 *(Langford, February 7, 2006)*, 618,801 *(Urrutia and Ribeiro, February 10 2006)*, 609788 *(Araujo, Urrutia, and Ribeiro, June 21, 2007)*, 598123 *(Goerigk, Hoshino, Kawarabayashi, and Westphal, Aug 27, 2013)*.
  Cota Inferior: 560,697 *(Trick, June 22, 2007)*.

- 🏆 NFL30 (30 teams):
  Solución Factible: 847,011 *(Di Gaspero and Schaerf, January 27 2006)*, 740,458 *(Urrutia and Ribeiro, February 10 2006)*, 739,697 *(Araujo, Urrutia, and Ribeiro, June 21, 2007)*.
  Cota Inferior: 688,875 *(Trick, June 22, 2007)*

- 🏆 NFL32 (32 teams):
  Solución Factible: 1,020,966 *(Di Gaspero and Schaerf, January 27 2006)*, 924,559 *(Urrutia and Ribeiro, February 10 2006)*, 914,620 *(Araujo, Urrutia, and Ribeiro, June 21, 2007)*.
  Cota Inferior: 836,031 *(Trick, June 22, 2007)*.

◦ Instancias de la Galaxia: estas instancias usan distancias tridimensionales que surgen de tomar la distancia entre las estrellas (en años luz). Los nombres se dan en base a la constelación en la cual aparece la estrella.

  ⋆ Galaxy 4:
  Solución Factible: 416 *(Uthus, Riddle, and Guesgen June 28, 2009)*
  Cota Inferior: 416 *(Uthus, Riddle, and Guesgen June 28, 2009)*

⋆ Galaxy 6:
Solución Factible: 1365 *(Uthus, Riddle, and Guesgen June 28, 2009)*
Cota Inferior: 1365 *(Uthus, Riddle, and Guesgen June 28, 2009)*

⋆ Galaxy 8:
Solución Factible: 2373 *(Uthus, Riddle, and Guesgen June 28, 2009)*
Cota Inferior: 2373 *(Uthus, Riddle, and Guesgen June 28, 2009)*

⋆ Galaxy 10:
Solución Factible: 4554 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 4535 *(Langford July 28, 2009).*
Cota Inferior: 4443 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 4535 *(Uthus, Riddle, and Guesgen December 25, 2009).*

⋆ Galaxy 12:
Solución Factible: 7357 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 71
97 *(Langford July 28, 2009)* Cota Inferior: 7034 *(Uthus, Riddle, and Guesgen January 6, 2010).*

⋆ Galaxy 14:
Solución Factible: 11412 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 11047 *(Langford August 1, 2009)*, 10918 *(Langford September 14, 2009)*
Cota Inferior: 10255 *(Uthus, Riddle, and Guesgen January 29, 2010)*

⋆ Galaxy 16:
Solución Factible: 16023 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 15099 *(Langford August 4, 2009)*, 14982 *(Langford September 20, 2009)*, 14900 *(Langford April 6, 2010)*
Lower Bound: 13619 *(independent).*

⋆ Galaxy 18:
Solución Factible: 22467 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 21571 *(Langford September 22, 2009)*, 21310 *(Langford December 18 2009)*, 20907 *(Langford March 13 2010)*, 20845 *(Hirano, Abe and Imahori May 21, 2018)*
Cota Inferior: 19050 *(independent)*

⋆ Galaxy 20:
Solución Factible: 29050 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 26855 *(Langford September 25, 2009)*, 26450 *(Langford December 27, 2009)*, 26289 *(Langford March 18, 2010).*
Cota Inferior: 23738 *(independent)*

⋆ Galaxy 22:
Solución Factible: 39025 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 37821 *(Langford September 25, 2009)*, 36296 *(Langford January 2, 2010)*, 36152 *(Langford March 7, 2010)*, 35767 *(Westphal August 23, 2010)*, 35516 *(Hosoe and Imahori, March 9, 2011)*, 35467 *(Goerigk and Westphal, Feb 9, 2012)*, 35095 *(Hoshino and Kawarabayashi, September 28, 2012)*, 35014 *(Hoshino and Kawarbayashi, November 29, 2012)*, 33901 *(Goerigk, Hoshino, Kawarabayashi, and Westphal, Aug 27, 2013).*
Cota Inferior: 31461 *(independent).*

⋆ Galaxy 24:
Solución Factible: 51056 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 48967 *(Langford October 2 2009)*, 48434 *(Langford January 11, 2010)*, 45910 *(Westphal August 23, 2010)*, 45728 *(Hosoe and Imahori, March 9, 2011)*, 45671 *(Goerigk and Westphal, December 23, 2011)*, 45610 *(Goerigk and Westphal, Feb 9, 2012)*, 45554 *(Hosoe and Imahori, Feb 24, 2012)*, 44647 *(Abe and Imahori, February 1, 2018)*, 44526 *(Hirano, Abe and Imahori May 21, 2018)*
Cota Inferior: 41287 *(independent)*

⋆ Galaxy 26:
Solución Factible: 66334 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 63493 *(Langford October 4 2009)*, 62613 *(Langford January 22, 2010)*, 60962 *(Westphal August 23, 2010)*, 59871 *(Goerigk and Westphal, December 23, 2011)*, 58968 *(Goerigk and Westphal, Feb 9, 2012)*
Cota Inferior: 53802 *(independent)*

⋆ Galaxy 28:
Solución Factible: 87109 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 83911 *(Langford October 4, 2009)*, 81911 *(Langford February 16, 2010)*, 77577 *(Westphal August 23, 2010)*, 77381 *(Goerigk and Westphal, December 23, 2011)*, 77214 *(Goerigk and Westphal, Feb 9, 2012)*, 77090 *(Hosoe and Imahori, Feb 24, 2012)*, 76749 *(Hoshino and Kawarabayashi,*

September 28, 2012), 76518 *(Hoshino and Kawarbayashi, November 29, 2012)*, 75276 *(Goerigk, Hoshino, Kawarabayashi, and Westphal, Aug 27, 2013)*
Cota Inferior: 69992 *(independent)*

⋆ Galaxy 30:
Solución Factible: 110237 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 104551 *(Langford October 24, 2009)*, 101774 *(Langford March 7, 2010)*, 96979 *(Westphal August 23, 2010)*, 96765 *(Hosoe and Imahori, March 9, 2011)*, 96712 *(Goerigk and Westphal, December 23, 2011)*, 96710 *(Goerigk and Westphal, Feb 9, 2012)*, 96580 *(Hosoe and Imahori, Feb 24, 2012)*, 95334 *(Abe and Imahori, February 1, 2018)*, 95158 *(Hirano, Abe and Imahori May 21, 2018)*
Cota Inferior: 88831 *(independent)*

⋆ Galaxy 32:
Solución Factible: 136253 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 127488 *(Langford October 27, 2009)*, 123510 *(Langford April 6, 2010)*, 120683 *(Westphal August 23, 2010)*, 120021 *(Goerigk and Westphal, December 23, 2011)*, 119996 *(Goerigk and Westphal, Feb 9, 2012)*, 119749 *(Abe and Imahori, February 1, 2018)*, 119665 *(Hirano, Abe and Imahori May 21, 2018).*
Cota Inferior: 108374 *(independent)*

⋆ Galaxy 34:
Solución Factible: 168721 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 162390 *(Langford October 27, 2009)*, 153530 *(Langford May 9, 2010)*, 147835 *(Westphal August 23, 2010)*, 147742 *(Hosoe and Imahori, March 9, 2011)*, 146982 *(Goerigk and Westphal, December 23, 2011)*, 147612 *(Goerigk and Westphal, Feb 9, 2012)*, 146792 *(Hosoe and Imahori, Feb 24, 2012)*, 145165 *(Hoshino and Kawarbayashi, November 29, 2012)*, 143298 *(Goerigk, Hoshino, Kawarabayashi, and Westphal, Aug 27, 2013).*
Cota Inferior: 133976 *(independent)*

⋆ Galaxy 36:
Solución Factible: 207117 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 187180 *(Langford November 24 2009)*, 177090 *(Langford July 3, 2010)*, 173827 *(Westphal August 23, 2010)*, 173640 *(Hosoe and Imahori, March 9, 2011)*, 173532 *(Goerigk and Westphal, Feb 9, 2012)*, 173458 *(Hosoe*

and Imahori, Feb 24, 2012), 171846 *(Abe and Imahori, February 1, 2018)*, 169387 *(Hirano, Abe and Imahori May 21, 2018)*
Cota Inferior: 158549 *(independent).*

⋆ Galaxy 38:
Solución Factible: 253279 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 227778 *(Langford November 24 2009)*, 214546 *(Langford, July 3, 2010)*, 210787 *(Westphal August 23, 2010)*, 209463 *(Hosoe and Imahori, March 9, 2011)*, 205725 *(Goerigk and Westphal, December 23, 2011)*, 204980 *(Goerigk and Westphal, Feb 9, 2012).*
Cota Inferior: 189126 *(independent)*

⋆ Galaxy 40:
Solución Factible: 304689 *(Uthus, Riddle, and Guesgen June 28, 2009)*, 281082 *(Langford November 24, 2009)*, 258899 *(Langford July 3 2010)*, 249230 *(Westphal August 23, 2010)*, 249002 *(Hosoe and Imahori, March 9, 2011)*, 247207 *(Goerigk and Westphal, December 23, 2011)*, 247017 *(Goerigk and Westphal, Feb 9, 2012)*, 245052 *(Hoshino and Kawarbayashi, November 29, 2012)*, 241908 *(Goerigk, Hoshino, Kawarabayashi, and Westphal, Aug 27, 2013).*
Cota Inferior: 226820 *(independent).*

4.2 Time Relaxed Round Robin Tournament Problem (TRRR)

4.2.1 Time Relaxed Single Round Robin Tournament

En este caso, la cantidad de intervalos de tiempo disponibles para utilizar como fechas es mayor que la cantidad mínima necesaria de fechas. Tiene carac-

terísticas diferentes comparándolo con la versión *time constrained*.

Los problemas de torneos TRSRR [16] tienen la estructura genérica del típico torneo de round robin: cada equipo se enfrenta con todos los demás una cantidad fija de veces (en este caso, una vez). Por lo dicho anteriormente, es necesario definir el parámetro de la cantidad de días disponibles para partidos. Se puede asumir que ese número sea el doble. O sea que, si n es la cantidad de equipos, luego, habría $2 * (n-1)$ intervalos de tiempo disponible si un equipo debe jugar $n-1$ partidos.

Si hay $2n$ equipos, se considera un grafo completo K_{2n} con $2n$ nodos. Como ya sabemos, cada nodo representa un equipo. Cada nodo tiene una arista que lo conecta a todos los demás nodos, lo cual representa un partido. Entonces, cada nodo tiene $2n-1$ nodos conectados con él. Necesitaríamos $2n-1$ colores para distinguir todas estas aristas. Si particionamos todas las aristas no adyacentes con el mismo color, obtenemos los 1-factores $F_1, ..., F_{2n-1}$ que corresponden al *schedule* de cada ronda $r = 1, ..., 2n-1$, y la dirección representa la localía. Entonces, se usa la llamada 1-factorización $(F_2, ..., F_n)$ canónica para construir un *schedule* con cantidad mínima de breaks. Los factores F_i se refiere a los conjuntos de aristas: $F_i = \{\{2n, i\}\} \cup \{\{i+k, i-k\} \,|k = 1, ..., n-1\}$

Los números $i+k$ e $i-k$ se toman módulo $2n-1$ como uno de los números $1, ..., 2n-1$. Si no hay restricciones en cuanto a la cantidad de partidos consecutivos que un equipo puede jugar, se puede usar la 1-factorización canónica para generar un *schedule* para el torneo TRRR, que tiene cantidad mínima de breaks.

Como ya sabemos, el *schedule* de un torneo TRTTP se puede modelar con *Programación Entera* o, también, con *Constraint Programming*.

Veamos el modelo de TRSRR con programación entera:

Se emplean $2n(n-1)^2$ variables y $\frac{5n(n-1)}{2}$ valores. Sea $T = \{t_1, ..., t_n\}$ un conjunto de n equipos y $D = \{D_k, k = 1, 2, .., 2(n-1)$ si n es par y $k = 1, 2, ..., 2n$ si n es impar $\}$

Cada equipo juega contra todo el resto exactamente una vez (de local o de visitante):

$$\sum_{d \in D}(x_{ijd} + x_{jid}) = 1 \qquad \forall i, j \in T, i < j \qquad (3.1)$$

Cada equipo juega como mucho un partido (de local o visitante) por ronda:

$$\sum_{j \in T \setminus \{i\}} (x_{ijd} + x_{jid}) \leq 1 \qquad \forall i \in T, d \in D \tag{3.2}$$

Las variables son binarias:

$$x_{ijd} \in \{0, 1\} \qquad \forall i, j \in T, i \neq j, d \in D \tag{3.3}$$

$$\sum_{i \in S, j \notin S} x_{ijd} \geq 1 \qquad \forall S \subset T,\ |S|\ impar \tag{3.4}$$

Si se suma sobre todos los subconjuntos S, entonces esta restricción nos asegura que cada equipo juegue exactamente una vez en cada ronda. El tema es que en este caso, no queremos forzar a que todos los equipos jueguen exactamente un partido cada día, con lo cual, no podríamos usar esta restricción.

4.2.2 Completar schedules parciales

En la práctica, se suele completar *schedules* parcialmente armados, en vez de crear uno de cero. Esto se debe a que hay muchos requerimientos por parte de fuentes externas como, por ejemplo, las emisoras de televisión. Es común que pidan que se reserven los partidos más importantes (los más atractivos, con los equipos más fuertes o rivales) en fechas específicas. Todas estas fechas reservadas, junto con otros requerimientos, componen el *schedule* parcialmente armado.

No es nada fácil completar un *schedule* parcial. De hecho, Easton [7] (2001) probó que no hay algoritmo de tiempo polinomial que complete un *schedule* de un torneo time constrained round robin parcial. Lamentablemente, no se puede probar formalmente la complejidad de este problema, se estima que es NP-hard.

4.2.3 Problema de Optimización

En la práctica, obtener un *schedule* factible no es el único requerimiento. Es bastante común que haya uno o más objetivos además de las muchas restricciónes adicionales. La programación de un schedule deportivo debe estar hecha por un agente externo dado que cada club tiene sus intereses específicos que pueden afectar a otro club. Con lo cual, una liga deportiva puede ser considerada como un "proveedor" en una competencia deportiva. Mientras que podríamos también

considerar a los fanáticos que van a presenciar el partido en el estadio, o en sus casas, y a los canales de televisión que transmiten el partido como "clientes". Más aún, los clubes deportivos generan ganancias a través de la comida y productos que venden durante estos eventos deportivos, vendiendo *merchandising* como gorras, camisetas, etc. Por eso, los clubes tendrían mayor ganancia si el schedule es bueno, ya que tendrían menores costos. Entonces, si consideramos que cada partido tiene un costo relacionado con los gastos que implica para los equipos, obtenemos un problema de optimización de un torneo TRSRR. Lo definimos formalmente de la siguiente manera:

Dado un conjunto de equipos T, $|T| = n$, y un set de días disponibles D, $|D| = 2 * (n-1)$, cada tripla $(i, j, d) \in T \times T \times D, i \neq j$ representa el partido de i contra j (i de local) en el día d. El costo $c_{i,j,d}$ está dado para cada partido. Una solución factible para el problema del torneo TRSRR corresponde a un conjunto de $\frac{n(n-1)}{2}$ triplas tales que:

(*i*) *para cada par* $(i, j) \in T \times T, i < j$, *se elige exactamente una tripla de la forma (i,j,d) o (j,i,d) con* $d \in D$

(*ii*) *para cada par* $(i, b) \in T \times D$, *se elige como mucho una tripla de la forma (i,j,b) o (j,i,b) con* $j \in T \backslash \{i\}$

El problema es encontrar una solución factible teniendo la suma mínima de los costos de las triplas elegidas.

La primera condición, fuerza a cada equipo a enfrentarse a todos los demás exactamente una vez; y la segunda, restringe a cada equipo a jugar como mucho un partido por día. El torneo TRSRR se puede modelar con un modelo de programación entera empleando $2n(n-1)^2$ variables y $\frac{5n(n-1)}{2}$ valores:

Función objetivo para minimizar los costos de todos los partidos:

$$min \sum_{i \in T} \sum_{j \in T \backslash \{i\}} \sum_{d \in D} x_{ijd} c_{ijd} \tag{3.5}$$

Cada equipo juega contra todos los demás exactamente una vez:

$$\sum_{d \in D}(x_{ijd} + x_{jid}) = 1 \qquad \forall i, j \in T, i < j \qquad (3.6)$$

Cada equipo juega a lo sumo una vez por fecha:

$$\sum_{j \in T \setminus \{i\}}(x_{ijd} + x_{jid}) \leq 1 \qquad \forall i \in T, d \in D \qquad (3.7)$$

Las variables son binarias:

$$x_{ijd} \in \{0, 1\} \qquad \forall i, j \in T, i \neq j, d \in D \qquad (3.8)$$

La variable binaria x_{ijd} es igual a 1 si el partido (i, j, b) se lleva a cabo, 0 si no. Las restricciones (3.6) y (3.7) corresponden a (i) y (ii) respectivamente, mientras (3.5) representa el objetivo de minimizar el costo.

Briskorn [4] probó que la versión time constrained de este problema de optimización es NP-hard. Con lo cual podemos conjeturar que nuestro problema es también NP-hard.

4.2.4 Time Relaxed Double Round Robin Tournament (TRDRRT)

En este caso, los equipos juegan contra todos los demás, exactamente dos veces en todo el torneo. Como, en nuestro caso, los días en que se juegan partidos no están fijos, no vamos a tener en cuenta torneos espejados, por ahora.

En la práctica es común que la liga requiera un torneo basado en rondas si tiene un *schedule* de múltiples rondas. Por ejemplo, muchas ligas tienen eventos "all-star" en los cuales los mejores deportistas de ese torneo disputan uno o varios partidos, fuera del *schedule* del torneo regular. Luego, es lógico haber completado una ronda antes de dicho evento. Vamos a examinar el torneo basado en rondas primero, y luego, veremos el torneo en general. Para ambos, veremos el modelo de programación entera que se usó.

4.2.5 Round-based TRDRRT

En el caso del torneo double round robin time constrained, la cantidad de días para la primera ronda es igual a la cantidad de días para la segunda ronda. En el caso time relaxed, necesitamos definir los días de partido para cada ronda. Usamos D_1 para representar el conjunto de días de la primera ronda, D_2 para representar el conjunto de días de la segunda ronda, D para el conjunto de días de ambas: $D_1 + D_2 = D$.

La definición del problema de optimización en este caso es similar a la anterior:

Dado un conjunto de equipos T, $|T| = n$*, y un set de días disponibles D.* $|D| = 4 * (n-1)$ *el conjunto de días se divide en dos subconjuntos* D_1 *y* D_2*. Cada tripla* $(i, j, d) \in T \times T \times D, i \neq j$ *representa un partido del equipo i contra el equipo j en el estadio del equipo i en el día d. Por cada partido se da un costo* $c_{i,j,d}$*.*

Una solución factible para el problema del Time Relaxed round-based double Round Robin Tournament corresponde a un conjunto de $n(n-1)$ *triplas tales que:*

(i) *para cada par* $(i, j) \in T \times T, i < j$*, se elige exactamente una tripla de la forma (i,j,d) o (j,i,d) con* $d \in D_1$

(ii) *para cada par* $(i, j) \in T \times T, i \neq j$*, se elige exactamente una tripla de la forma (i,j,d) o (j,i,d) con* $d \in D_2$

(iii) *para cada par* $(i, d) \in T \times D$*, se elige como mucho una tripla de la forma (i,j,d) o (j,i,d) con* $j \in T \backslash \{i\}$

El problema es encontrar una solución factible teniendo la suma mínima de los costos de las triplas elegidas.

Las variables son las mismas que se usaron en el modelo del problema del torneo single round robin:

Función objetivo para minimizar los costos de todos los partidos:

$$min \sum_{i \in T} \sum_{j \in T \backslash \{i\}} \sum_{d \in D} x_{ijd} c_{ijd} \tag{3.9}$$

Cada equipo juega contra todos los demás exactamente una vez en la primera ronda:

$$\sum_{d \in D_1}(x_{ijd} + x_{jid}) = 1 \qquad \forall i, j \in T, i < j \tag{3.10}$$

Cada equipo juega una exactamente una vez contra todos los demás equipos de local durante toda la temporada:

$$\sum_{d \in D} x_{ijd} = 1 \qquad \forall i, j \in T, i \neq j \tag{3.11}$$

Cada equipo juega a lo sumo una vez por fecha:

$$\sum_{j \in T \setminus \{i\}}(x_{ijd} + x_{jid}) \leq 1 \qquad \forall i \in T, d \in D \tag{3.12}$$

Las variables son binarias:

$$x_{ijd} \in \{0, 1\} \qquad \forall i, j \in T, i \neq j, d \in D \tag{3.13}$$

La complejidad de este problema, depende de la del anterior. Entonces, podemos conjeturar que este problema tiene complejidad NP-hard.

4.2.6 Torneo Time Relaxed Double Round Robin General

Dado un conjunto de equipos T, $|T| = n$, y un set de días disponibles D, $|D| = 4 * (n-1)$, cada tripla $(i, j, d) \in T \times T \times D, i \neq j$ representa el partido de i contra j (i de local) en el día d. El costo $c_{i,j,d}$ está dado para cada partido. Una solución factible para el problema del torneo TRgDRR corresponde a un conjunto de $n(n-1)$ triplas tales que:

(i) para cada par $(i, j) \in T \times T, i \neq j$, se elige exactamente una tripla de la forma (i,j,d) con $d \in D$

(ii) para cada par $(i, b) \in T \times D$, se elige como mucho una tripla de la forma (i,j,b) o (j,i,b) con $j \in T \setminus \{i\}$

El problema es encontrar una solución factible teniendo la suma mínima de los costos de las triplas elegidas.

La primera condición, fuerza a cada equipo a enfrentarse a todos los demás exactamente una vez de local durante toda la temporada; y la segunda, restringe

a cada equipo a jugar como mucho un partido por día.

Función objetivo para minimizar los costos de todos los partidos:

$$min \sum_{i \in T} \sum_{j \in T \setminus \{i\}} \sum_{d \in D} x_{ijd} c_{ijd} \tag{3.14}$$

Cada equipo juega contra todos los demás exactamente una vez:

$$\sum_{d \in D} x_{ijd} = 1 \qquad \forall i, j \in T, j \neq i \tag{3.15}$$

Cada equipo juega a lo sumo una vez por fecha:

$$\sum_{j \in T \setminus \{i\}} (x_{ijd} + x_{jid}) \leq 1 \qquad \forall i \in T, d \in D \tag{3.16}$$

Las variables son binarias:

$$x_{ijd} \in \{0, 1\} \qquad \forall i, j \in T, d \in D \tag{3.17}$$

La variable binaria x_{ijd} es igual a 1 si el partido (i, j, b) se lleva a cabo, 0 si no.

4.2.7 Problema del Torneo Time Relaxed r Round Robin General

Generalmente, se tiene un torneo r round robin cuando cada equipo juega contra todos los demás equipos r veces. Al igual que los torneos de dos rondas, tenemos el torneo basado en rondas y el general. Ambos similares a los problemas definidos anteriormente respectivamente. Asumimos que r es par. Definamos la versión general del problema:

Dado un conjunto de equipos T, $|T| = n$, y un set de días disponibles D, $|D| = 2r * (n - 1)$, cada tripla $(i, j, d) \in T \times T \times D, i \neq j$ representa el partido de i contra j (i de local) en el día d. El costo $c_{i,j,d}$ está dado para cada partido. Una solución factible para el problema del torneo TRgr-RR corresponde a un conjunto de $\frac{n(n-1)r}{2}$ triplas tales que:

(i) para cada par $(i, j) \in T \times T, i \neq j$, se elige exactamente una tripla de la forma (i,j,d) con $d \in D$

(ii) *para cada par $(i, b) \in T \times D$, se elige como mucho una tripla de la forma (i,j,b) o (j,i,b) con $j \in T \backslash \{i\}$*

El problema es encontrar una solución factible teniendo la suma mínima de los costos de las triplas elegidas.

La primera condición, fuerza a cada equipo a enfrentarse a todos los demás exactamente una vez de local durante toda la temporada; y la segunda, restringe a cada equipo a jugar como mucho un partido por día.

Función objetivo para minimizar los costos de todos los partidos:

$$min \sum_{i \in T} \sum_{j \in T \backslash \{i\}} \sum_{d \in D} x_{ijd} c_{ijd} \tag{3.18}$$

Cada equipo juega contra todos los demás exactamente $\frac{r}{2}$ veces durante toda la temporada:

$$\sum_{d \in D} x_{ijd} = \frac{r}{2} \qquad \forall i, j \in T, j \neq i \tag{3.19}$$

Cada equipo juega a lo sumo una vez por fecha:

$$\sum_{j \in T \backslash \{i\}} (x_{ijd} + x_{jid}) \leq 1 \qquad \forall i \in T, d \in D \tag{3.20}$$

Las variables son binarias:

$$x_{ijd} \in \{0, 1\} \qquad \forall i, j \in T, d \in D \tag{3.21}$$

La variable binaria x_{ijd} es igual a 1 si el partido (i, j, b) se lleva a cabo, 0 si no. Por lo dicho anteriormente, se conjetura que el torneo r round robin es NP-hard.

4.3 Un Algoritmo Mejorado de Aproximación para el Traveling Tournament Problem con Longitud Máxima 2

A pesar de la investigación intensiva que se realizó de este problema durante las últimas dos décadas, la mayoría de las instancias con más de 10 equipos en los benchmarks más conocidos no fue resuelta aún. Los autores M Xiao, S Kou [27]

desarrollaron un algoritmo de aproximación práctico para este problema con restricciones tales que se permitan como mucho dos partidos consecutivos de local o dos partidos consecutivos de visitante. Este algoritmo, que genera *schedules* factibles basados en matching mínimos perfectos del grafo subyacente, no sólo mejora la relación de aproximación de $1 + \frac{16}{n}$ a $1 + \frac{4}{n}$, sino que también tiene performance experimental muy buena. Aplicando los *schedules* resultantes a conjuntos de referencia conocidos se mejoraron todos los resultados previos conocidos de instancias con n siendo múltiplo de 4 en un 3% o 10%. Notar que la relación de aproximación de un algoritmo es la relación entre el resultado obtenido por el algoritmo y el costo óptimo o ganancia.

En este caso, a las restricciones básicas conocidas para el TTP genérico, le vamos a agregar una restricción más: ningún equipo puede tener un break que dure más de k partidos. Así, quedaría definido el TTP-k.

En este caso, tenemos la misma cantidad de partidos a jugar que fechas disponibles para ellos. Es decir, es un Time Constrained TTP.

El entero k en el imput define el tradeoff entre la distancia viajada y la longitud de los breaks. Para el caso $k = \infty$, no hay restricciones para la cantidad de partidos de local (o visitante) seguidos y a un equipo se le puede asignar una distancia para viajar tal como la del *road trip* de ciudades del vendedor ambulante (Traveling Salesman Problem). El TTP-k puede ser visto como una variante del TSP.

Lo que sigue se enfocará en el TTP-2. Este problema fue mencionado por primera vez por Campbell y Chen [4], quienes mencionamos anteriormente, ya que armaron el *schedule* de una conferencia de básquet de 10 equipos. En el TTP-2, todos los *road trips* consisten en un equipo sólo o en pares de equipos. En la práctica, es razonable que cada equipo tenga breaks de a lo sumo dos partidos. En un *schedule*, se espera que los breaks se alternen (entre local y visitante) tanto como se pueda para cada equipo. Se puede ver que el *schedule* perfecto con $k = 1$ no se puede alcanzar. Es natural considerar, entonces el TTP-2. Una gran contribución al algoritmo de aproximación para TTP-2 debe atribuirse a Thielen y Westphal [18]. Ellos fueron los primeros en dar la relación del algoritmo de aproximación $\frac{3}{2} + O(\frac{1}{n})$ y, luego, la mejoraron a $1 + \frac{16}{n}$ para el caso en que $n \geq 12$ y n es múltiplo de 4.

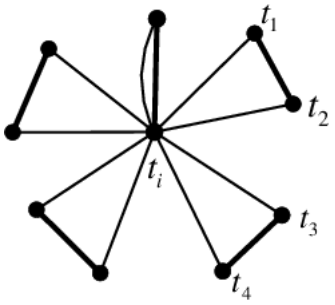
El algoritmo que se presentará a continuación genera una solución factible para el TTP-2 para n múltiplo de 4 con una distancia a viajar de a lo sumo $(1+\frac{2}{(n-2)}+\frac{2}{n})$-

veces la distancia óptima, mejorando la relación de aproximación previa de $1 + \frac{16}{n}$.

4.3.1 Una Cota inferior

Como se dijo anteriormente, la mayoría de las cotas inferiores del TTP se obtienen a través de una técnica de relajación llamada "independiente". Se trata de computar la distancia mínima de viaje "factible" para cada equipo independientemente y, luego, sumar todas juntas para obtener la ya mencionada *Cota Inferior Independiente*. Acá, "factible" significa que los viajes cumplen las tres condiciones de la definición del TTP-k.

También, Campbell y Chen [4] obtuvieron la primera cota inferior independiente para el TTP-2. Se puede calcular encontrando un matching mínimo perfecto en un grafo completo no dirigido G del conjunto de equipos con el peso de las aristas siendo la distancia entre los estadios de dos equipos. Gracias a la desigualdad triangular, sabemos que lo que vaya a viajar un equipo, de forma óptima y factible, conlleva a lo sumo un *road trip* para un sólo equipo y el resto *road trip*s de pares de equipos. De acuerdo a la definición del problema, el número n de equipos es par. Entonces, cada equipo contiene exactamente un *road trip* que consiste de un sólo partido y el resto son *road trips* que contienen pares de equipos cada una. El grafo del itinerario para t_i es el siguiente:



Cada equipo t_1 debe viajar hacia o desde el estadio de cada equipo al menos una vez. La longitud total de todas las lineas claras es constante, es la distancia total desde el estadio del equipo t_i a las ciudades de todos los demás equipos, la cual también se denota D_i. Tenemos que $D_i = \sum_{j \neq i} D_{i,j}$.

Las lineas oscuras en la figura forman un matching perfecto de G. Se usa M para denotar un matching perfecto mínimo de G (un matching perfecto con peso total de las aristas mínimo) y usar D_M para denotar el peso total de las aristas de

M. Observar que la distancia viajada del equipo t_i es al menos $LB_i = D_i + D_M$, la cual llamamos Cota Inferior Independiente para el equipo t_i. Se una D_G para denotar la suma de los pesos de todas las aristas de G. Una cota inferior para el TTP, obtenida a través de la suma de todas las cotas inferiores independientes de cada equipo, es:
$LB = \sum_{i=1}^{n} LB_i = \sum_{i=1}^{n}(D_i + D_M) = 2D_G + nD_M$ (1)

Si pudiéramos encontrar un *schedule* de un torneo factible tal que todos los equipos alcancen la cota inferior independiente simultáneamente, el TTP estaría resuelto óptimamente. Pero eso es imposible para un *schedule* factible. Cabe mencionar que incluso para $k \geq 3$, computar la cota inferior independiente es NP-hard, pues incluye el problema de encontrar un packing (un tipo de partición) de k caminos en un grafo.

4.3.2 Técnicas de Construcción

No es trivial obener un *schedule* factible para el TTP-2, aún sin considerar la distancia viajada. La *"Expander construction"* (construcción expansiva) es un método efectivo usado para construir *schedules* factibles para TTP-3. Se modificará este método para TTP-2 para construir una solución inicial. Luego de obtener un *schedule* factible, se usan tecnicas basadas en matchings perfectos mínimos para arreglar el orden de los equipos y, luego, se puede obtener una solución con la distancia viajada bastante cerca de la cota inferior independiente.

Para hacer que la distancia viajada sea más corta, se espera que en un *road trip* de un partido (un viaje de visitante de sólo un partido) sea un par en un perfect matching mínimo de G. Luego de armar el *schedule* para los pares, "expandimos" reemplazando cada par con dos equipos originales para obtener el *schedule* final.

La construcción consta de dos pasos. El primer paso consiste en crear un torneo single round robin U_m para $m = \frac{n}{2}$ equipos (cada uno de los cuales representa un par de equipos originalmente). El segundo paso es expandir U_m a un torneo double round robin Z_n de n equipos. Notar que la construcción sólo funciona para $m = \frac{n}{2}$ par, es decir, $n \equiv 0(4)$. A partir de ahora, asumiremos que m es par.

Paso 1: Construir un torneo U_m single round robin

El torneo U_m single round robin de m equipos se construye usando una variación del Método del Círculo Modificado. Se usará $\{u_1, u_2, ..., u_{m-1}, x\}$ para denotar los m equipos. Cada equipo juega contra los demás m-1 equipos en m-1

intervalos de tiempo tal que para cada1 $\leq i \leq m-1$, el equipo u_i juega contra el equipo u_j en un intervalo de tiempo r tal que $r - i \equiv j - 1(m-1)$, donde interpretamos el caso en que u_i juega contra sí mismo como que juega contra x en ese intervalo de tiempo. Esta asignación garantiza un *schedule* factible, es decir que cada uno de los m equipos juega contra todos los demás en cada uno de los m-1 intervalos de tiempo.

La construcción no termina acá. Se asigna un equipo local y uno visitante para cada partido que no incluye al equipo x: para cada $1 \leq i \leq \frac{m}{2}$, u_i juega sólo partidos de visitante hasta que se encuentra con el equipo x, antes de terminar los partidos restantes de local*; para cada $\frac{m}{2} \leq i \leq m-1$, tenemos un caso opuesto, donde u_i juega sólo de local los partidos hasta que se encuentra con c, antes de terminar los partidos restantes de visitante. En la siguiente tabla, se ve el *schedule* con $m = 8$, donde los items en negrita indican que los equipos correspondientes (los de la izquierda de la tabla) son locales en este partido.

Paso 2: Construir un torneo double round robin Z_n

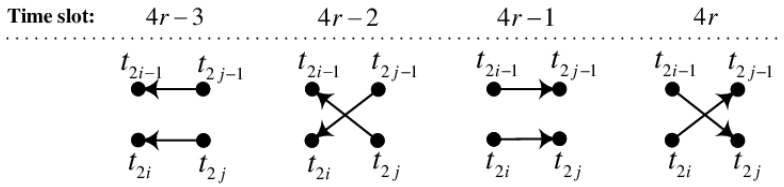
Tenemos cuatro subpasos para contruir este torneo Z_n de n equipos desde el torneo single round robin U_m de $m = \frac{n}{2}$ equipos. Recordemos que cada equipo u_i en U_m es representado por un par de equipos originales en el torneo. Luego, reemplazamos u_i (donde x se interpreta como u_m) con dos equipos originales $\{t_{2i-1}, t_{2i}\}$ en este paso. Entonces, el conjunto de n equipos originales en Z_n se denota $\{t_1, t_2, ..., t_{n-1}, t_n\}$.

Acá tenemos la construcción del simple round robin para $m = 8$:

	1	2	3	4	5	6	7
u_1	ⓧ	$\boldsymbol{u_2}$	$\boldsymbol{u_3}$	$\boldsymbol{u_4}$	$\boldsymbol{u_5}$	$\boldsymbol{u_6}$	$\boldsymbol{u_7}$
u_2	u_7	u_1	ⓧ	$\boldsymbol{u_3}$	$\boldsymbol{u_4}$	$\boldsymbol{u_5}$	$\boldsymbol{u_6}$
u_3	u_6	u_7	u_1	u_2	ⓧ	$\boldsymbol{u_4}$	$\boldsymbol{u_5}$
u_4	u_5	u_6	u_7	u_1	u_2	u_3	ⓧ
u_5	$\boldsymbol{u_4}$	ⓧ	u_6	u_7	u_1	u_2	u_3
u_6	$\boldsymbol{u_3}$	$\boldsymbol{u_4}$	$\boldsymbol{u_5}$	ⓧ	u_7	u_1	u_2
u_7	$\boldsymbol{u_2}$	$\boldsymbol{u_3}$	$\boldsymbol{u_4}$	$\boldsymbol{u_5}$	$\boldsymbol{u_6}$	ⓧ	u_1
x	u_1	u_5	u_2	u_6	u_3	u_7	u_4

En U_m, un partido en el último intervalo de tiempo se llama *último partido* y un partido que no se juega en el último intervalo de tiempo se llama *partido normal*. Para construir Z_n, distinguimos cuatro tipos de juegos en U_m de acuerdo a si el partido es un último partido o no y si involucra a x o no.

- Caso 1: partidos normales que no involucran a x:
  Consideramos un partido en U_m, donde un equipo local u_i juega contra un equipo visitante u_j en un intervalo de tiempo r ($1 \leq i, j \leq m-1$ y $1 \leq r \leq m-2$). Se expande este partido a $2 \times 4 = 8$ partidos en cuatro intervalos de tiempo consecutivos en Z_n. Los cuatro intervalos de tiempo correspondientes son desde $4r-3$ a $4r$. Recordar que u_i será reemplazado por $\{t_{2i-1}, t_{2i}\}$ y que u_j será reemplazado por $\{t_{2j-1}, t_{2j}\}$. La figura muestra cómo los cuatro equipos jugarán en los cuatro intervalos de tiempo, donde un arco de a a b significa que el equipo a juega de visitante contra el equipo b.

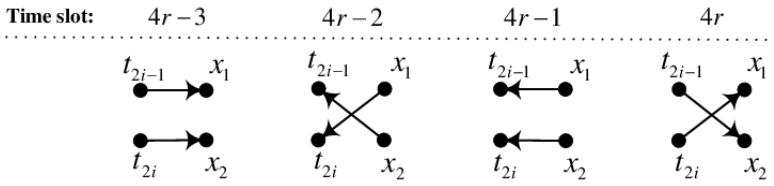


  Notar que en este esquema, cada uno de los $\{t_{2i-1}, t_{2i}\}$ tiene un *road trip* de dos equipos en $\{t_{2j-1}, t_{2j}\}$, y también, cada uno de los $\{t_{2j-1}, t_{2j}\}$ tiene un *road trip* de dos equpos en $\{t_{2i-1}, t_{2i}\}$. Lo que es más, no hay conflicto en asignar los partidos en Z_n correspondientes a todos los partidos del Caso 1 en U_m, es decir, las tres condiciones del TTP-k se mantienen.

- Caso 2: partidos normales que involucran a x:
  Consideremos un partido en U_m, donde el equipo u_i juega contra el equipo x en el intervalo de tiempo r ($1 \leq i \leq m-1$ y $1 \leq r \leq m-2$). Para que sea más fácil de representar, usamos x_1 y x_2 para denotar los dos equipos en Z_n correspondientes a x, es decir, $x_1 = t_n - 1$ y $x_2 = t_n$. También expandimos este partido a $2 \times 4 = 8$ partidos en cuatro intervalos de tiempo consecutivos en Z_n. De todas formas, las expansiones son diferentes de acuerdo a si el intervalo de tiempo r es par o impar.

  En un intervalo de tiempo r, un equipo u_i con $1 \leq i \leq \frac{m}{2}$ juega contra el equipo x. Asignamos los partidos entre los cuatro equipos $\{t_{2i-1}, t_{2i}, x_1, x_2\}$

en los intervalos de tiempo de $4r-3$ a $4r$.



En la siguiente tabla, se ven los correspondientes 16 items en Z_n determinados por los partidos de la figura anterior.

	$4r-3$	$4r-2$	$4r-1$	$4r$
t_{2i-1}	x_1	$\boldsymbol{x_2}$	$\boldsymbol{x_1}$	x_2
t_{2i}	x_2	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	x_1
x_1	$\boldsymbol{t_{2i-1}}$	t_{2i}	t_{2i-1}	$\boldsymbol{t_{2i}}$
x_2	$\boldsymbol{t_{2i}}$	t_{2i-1}	t_{2i}	$\boldsymbol{t_{2i-1}}$

En el caso en que el intervalo de tiempo r sea par, el equipo u_i con $\frac{m}{2}+1 \leq i \leq m-1$ juega contra el equipo x. El *schedule* es casi igual que en la tabla anterior. Sólo necesitamos intercambiar la asignación de localías en cada partido.

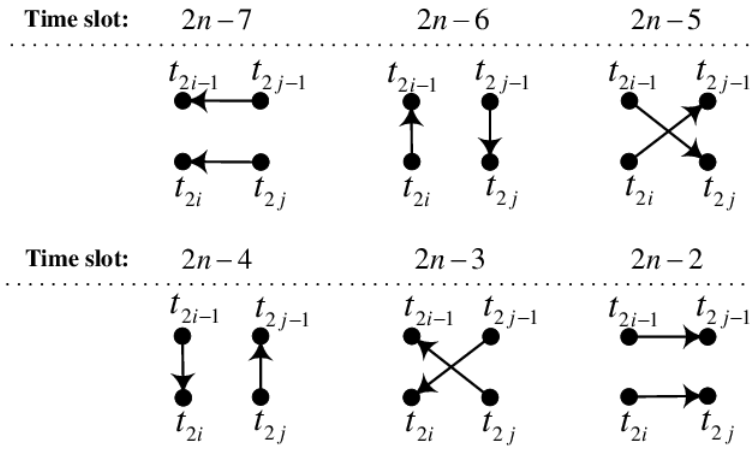
Para el Caso 2, se usa una estrategia de construcción diferente del Caso 1 tal que se pueda satisfacer la condición de "a lo sumo k". De este *schedule*, se puede ver que cada uno de los $\{t_{2i-1}, t_{2i}\}$ tiene dos *road trips* de un solo equipo, que son x_1 y x_2, y cada uno de los $\{x_1, x_2\}$ tiene un *road trip* de dos equipos en $\{t_{2i-1}, t_{2i}\}$.

Luego de expandir los partidos del Caso 1 y 2, sólo resta expandir los últimos partidos del último intervalo de tiempo en U_m. Si expandimos los últimos partidos de acuerdo a las reglas del Caso 1 y 2, no tendríamos ningún problema "superficial". Pero, luego de esto, todavía hay dos partidos que no fueron asignados para cada equipo, que son entre los dos equipos t_{2i-1} y t_{2i} que corresponden a u_i en U_m. Estos dos partidos no pueden ser asignados a dos intervalos de tiempo consecutivos dada la condición de no repetir en la definición del TTP-k. Entonces, va a ser dificil encontrar un lugar para asignar estos dos partidos. Para resolver esto, la idea es expandir el último intervalo de tiempo en U_m en seis (en vez de cuatro) intervalos de tiempo en

Z_n, dos de los cuales son el esquema de los últimos dos partidos.

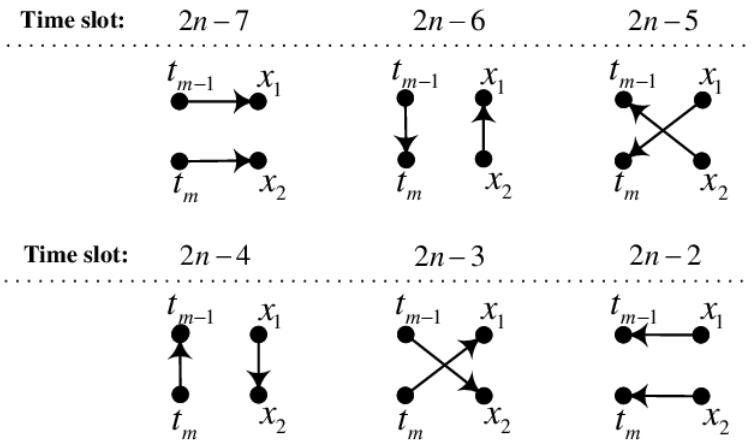
- Caso 3: últimos partidos que no involucran a x:

  Consideramos un partido en U_m, donde un equipo local u_i juega de local contra un equipo u_j que juega de visitante en el intervalo de tiempo $m-1$ $(1 \leq i, j \leq m-1)$. Expandimos este partido en $2 \times 6 = 12$ partidos en seis intervalos de tiempo consecutivos, de $2n-7$ a $2n-2$, en Z_n.



- Caso 4: últimos partidos que involucran a x:

  Luego del Caso 3, hay un solo partido en U_m que no se expandió. Es el partido donde el equipo $u_{\frac{m}{2}}$ juega contra x en un intervalo de tiempo $m-1$. Se expande este partido a $2 \times 6 = 12$ partidos en seis intervalos de tiempo consecutivos en Z_n de acuerdo a una estrategia similar a la figura anterior. Sólo necesitamos reeemplazar x_1 y x_2 por t_{2j-1} y t_{2j}, respectivamente, e intercambiar la designación de localías en cada partido de la figura anterior. La siguiente figura muestra los 12 partidos en los seis intervalos de tiempo.



	$2n-7$	$2n-6$	$2n-5$	$2n-4$	$2n-3$	$2n-2$
t_{m-1}	x_1	t_m	$\boldsymbol{x_2}$	$\boldsymbol{t_m}$	x_2	$\boldsymbol{x_1}$
t_m	x_2	$\boldsymbol{t_{m-1}}$	$\boldsymbol{x_1}$	t_{m-1}	x_1	$\boldsymbol{x_2}$
x_1	$\boldsymbol{t_{m-1}}$	$\boldsymbol{x_2}$	t_m	x_2	$\boldsymbol{t_m}$	t_{m-1}
x_2	$\boldsymbol{t_m}$	x_1	t_{m-1}	$\boldsymbol{x_1}$	$\boldsymbol{t_{m-1}}$	t_m

Es fácil ver que esta construcción puede implementarse en tiempo $O(n^2)$. En el *schedule* completo del torneo para Z_8, los intervalos de tiempo 1-4 y 5-8 para los equipos t_3 y t_4 corresponden al Caso 1, los intervalos de tiempo 1-4 para los equipos t_1 y t_2 corresponden al Caso 2, los intervalos de tiempo 9-14 para los equipos t_5 y t_6 corresponden al Caso 3 y los intervalos de tiempo 9-14 para los equipos t_3 y t_4.

4.3.3 Sschedule basado en Matchings Perfectos

La estrategia anterior provee un *schedule* factible del torneo para cualquier orden de los n equipos. Hay $n!$ permutaciones de los n equipos. Para minimizar la distancia total viajada, se ordenan los equipos de acuerdo a un matching perfecto mínimo M de G, donde G es un grafo completo no dirigido del conjunto de equipos con aristas pesadas representando la distancia entre los equipos.

Un orden los de equipos $\{t_1, t_2, ..., t_n\}$ es *consistente* con un matching perfecto mínimo M si para cada i impar, los equipos t_i y t_{i+1} están en un par en M, es decir, cada par de equipos correspondientes a un equipo en U_m es también un par en M. Hay muchos ordenes de los equipos consistentes con el matching M. Se muestra, aquí, un algoritmo para encontrar buenos órdenes de los equipos consistentes con M, lo cual puede arrojar *schedules* del torneo con buenas performances en la teoría y en la práctica. El algoritmo consta de cinco pasos.

- **Paso 1:** Construir el grafo completo no dirigido G y computar un matching perfecto mínimo M de G. Notar que G tiene n vértices y M tiene $\frac{n}{2}$ aristas.

- **Paso 2:** Construir otro grafo completo no dirigido H basado en G y M. El grafo H tiene $\frac{n}{2}$ vértices $\{u_{i_1}, u_{i_2}, ..., u_{i_m}\}$. Cada vértice u_i corresponde a una arista $t_{2i-1}t_{2i}$ en M (tambíen a un equipo en U_m). El peso de cada arista u_iu_j entre dos vértices u_i y u_j en H, que se denota $w_H(u_i, u_j)$, es el peso total de las cuatro aristas entre $\{t_{2i-1}, t_{2i}\}$ y $\{t_{2j-1}, t_{2j}\}$ en G.

- **Paso 3:** Encontrar un vértice, que se denota u_m, en H tal que el peso de todas las aristas incidentes en él sea mínimo.

- **Paso 4:** Encontrar un matching perfecto mínimo M_H. Decimos $M_H = \{u_1u_{m-1}, u_2u_{m-2}, ..., u_{\frac{m}{2}-1}u_{\frac{m}{2}+1}, u_{\frac{m}{2}}u_m\}$.

- **Paso 5:** Ordenar los n equipos de acuerdo a M_H. Obtenemos un orden $\{t_1, ..., t_n\}$ de los equipos tales que: t_{2i-1} y t_{2i} son dos equipos en la arista en M correspondiente a u_i para $1 \leq i \leq m$.

Este algoritmo computa principalmente dos matchings perfectos mínimos y corre en tiempo $O(n^3)$.

4.3.4 Análisis de la relación de aproximación

Mediante este análisis, se ven las ventajas de tomar estas permutaciones de los equipos.

Usamos A_i para denotar la distancia viajada del equipo t_i en nuestro *schedule*. La distancia total viajada en el torneo bajo nuestro *schedule* es $A_{ALL} = \sum_{i=1}^{n} A_i$. Para evaluar la performance de nuestro esquema, vamos analizar la relación $\frac{A_{ALL}}{LB}$. Sea $\Delta = A_{ALL} - LB$. Primero, comparamos A_i con la cota inferior independiente LB_i. Sea $\Delta_i = A_i - LB_i$. Se analiza Δ_i para diferentes casos de acuerdo al valor de i.

- **Caso (i).** Los equipos t_i con $1 \leq i \leq \frac{n}{2} - 2$:
  Estos equipos en el toreno double round robin Z_n son expandidos de las primeras $\frac{m}{2}$ líneas (equipos) en el torneo single round robin U_m. Miramos la línea del equipo t_i en Z_n. Para la parte expandida desde U_m del Caso 1 (partidos normales que no involucran a x), t_i tiene un *road trip* que consiste en dos equipos en una arista del matching M. Para la parte expandida de desde U_m del Caso 2 (partidos normales que involucran a x en intervalos de tiempo impares), t_i tiene dos *road trips* que consisten en un solo equipo de $x_1 = t_n - 1$ y $x_2 = t_n$ respectivamente. Para la parte expandida desde U_m del Caso 3 (últimos partidos que no involucran a x), t_i tiene un *road trip* que consiste en un solo equipo t_{ji} y un *road trip* que consta de dos equipos t'_i y $t_{j'i}$, donde t_i y $t_{i'}$ (respectivamente t_{ji} y $t_{j'i}$) corresponponden al mismo equipo u_q (respectivamente u_p) en U_m y $p + q = m$. Si comparamos el grafo del itinerario del equipo t_i con el itinerario óptimo para alcanzar la cota inferior independiente, se obtiene la siguiente desigualdad triangular:

  $$\Delta_i = (D_{i,n-1} + D_{i,n}) - D_{n-1,n} + (D_{i,j_i} + D_{j'_i,i'}) -$$

  $$(D_{i,i'} + D_{j_i,j'_i}) \leq (D_{i,n-1} + D_{i,n}) + (D_{i,j_i} + D_{i,j'_i}).$$

  Recordar que usamos $w_H(u_i, u_j)$ para denotar el peso de las aristas entre dos vertices u_i y u_j en H. Tenemos que:

  $$\sum_{i=1}^{\frac{n}{2}-2} \Delta_i \leq \sum_{i=1}^{\frac{m}{2}-1} w_H(u_i, u_m) + \sum_{i=1}^{\frac{m}{2}-1} w_H(u_i, u_{m-i}) \qquad (2)$$

- **Caso (ii).** Los equipos t_i con $i \in \{\frac{n}{2} - 1, \frac{n}{2}\}$:
  Estos equipos en Z_n son expandidos desde la línea $\frac{m}{2}$ en U_m. Hay sólo dos formas de expansión: Caso 1 y Caso 4. Análogamente al Caso (i), tenemos en itinerario de $t_{\frac{m}{2}-1}$. Y el grafo del itinerario para $t_{\frac{m}{2}}$es similar. Entonces, tenemos que

$\Delta_{\frac{n}{2}-1} = (D_{\frac{n}{2}-1,n} + D_{\frac{n}{2},n-1}) - (D_{\frac{n}{2}-1,\frac{n}{2}} + D_{n-1,n})$

$\leq (D_{\frac{n}{2}-1,n-1} + D_{\frac{n}{2}-1,n})$

y también $\Delta_{\frac{n}{2}} \leq (D_{\frac{n}{2},n-1} + D_{\frac{n}{2},n})$,

entonces, $\Delta_{\frac{n}{2}-1} + \Delta_{\frac{n}{2}} \leq w_H(u_{\frac{m}{2}}, u_m)$ (3)

- **Caso (iii).** Los equipos t_i con $\frac{n}{2}+1 \leq i \leq -2$:
  Estos equipos en Z_n son expandidos desde las líneas de la $\frac{m}{2}$ a la $m-1$ en U_m. Hay tres clases de expansiones: Caso 1, Caso 2 y Caso 3, donde las expansiones del Caso 2 están en intervalos de tiempo pares. Luego t_i tiene un *road trip* que consiste en dos equipos x_1 y x_2. Obtenemos, entonces, que

  $\Delta_i = (D_{i,j_i} + D_{j_i',i'}) - (D_{j_i,j_i'} + D_{j_i,j_i'})$

  $\leq (D_{i,j_i} + D_{i,j_i'})$,

  donde t_i y t_i' (respectivamente t_j y t_j') corresponden al mismo equipo u_q (respectivamente u_p) en U_m y $p+q=m$.

  Sumando sobre i en este caso, tenemos

  $\sum_{i=\frac{n}{2}+1}^{n-2} \Delta_i \leq \sum_{i=\frac{m}{2}+1}^{m-1} w_H(u_i, u_{m-i})$ (4)

- **Caso (iv).** Los equipos t_i con $i \in \{n-1, n\}$:
  Los dos últimos equipos en Z_n son expandidos desde la última línea de U_m. Hay dos tipos de expansiones que involucran a x: Caso 2 y Caso 4. Obtenemos que

  $\Delta_{n-1} = \sum_{j=\frac{m}{2}+1}^{m-1}((D_{n-1,2j-1} + D_{n-1,2j}) - D_{2j-1,2j}))$

  $+(D_{m-1,n-1} + D_{m,n}) - (D_{m-1,m} + D_{n-1,n})$

  $\leq \sum_{j=\frac{m}{2}+1}^{m-1}(D_{n-1,2j-1} + D_{n-1,2j}) + (D_{m,n-1} + D_{m-1,n-1})$

$= \sum_{j=\frac{m}{2}}^{m-1}(D_{n-1,2j-1} + D_{n-1,2j})$

y también, $\Delta_n = \sum_{j=\frac{m}{2}}^{m-1}(D_{n,2j-1} + D_{n,2j})$

Luego, $\Delta_{n-1} + \Delta_n \leq \sum_{j=\frac{m}{2}}^{m-1} w_H(u_m, u_j)$. (5)

Sumando (2), (3), (4) y (5), obtenemos

$\Delta = \sum_{i=1}^{n} \Delta_i$

$\leq \sum_{j=1}^{m-1} w_H(u_m, u_j) + 2\sum_{i=\frac{m}{2}-1}^{m-1} w_H(u_i, u_m - i)$

$+w_H(u_i, u_m - i)w_H(u_{\frac{m}{2}}, u_m)$

$\leq w_H(E(u_m)) + 2w_H(M_H)$ (6)

donde $w_H(E(u_m))$ es el peso total de todas las aristas incidentes en u_m en H y $w_H(M_H)$ es el peso de todas las aristas en el matching M_H. Sea D_H el peso de todas las aristas en H. Luego,

$D_H = D_E - D_M$ (7)

Dado que elegimos u_m como el vértice de H tal que el peso de todas las aristas incidentes en él es mínimo, sabemos que

$w_H(E(u_m)) \leq \frac{2}{m} D_H$ (8)

Notar que un grafo completo de m vértices puede ser particionado en $m-1$ matchings perfectos. Elegimos M_H como matchings perfectos de peso mínimo. Luego, tenemos que

$w_H(M_H) \leq \frac{1}{m-1} DH$ (9)

Por (1), (6), (7), (8), (9) y $n = 2m$, obtenemos

$\Delta = (\frac{2}{m} + \frac{2}{m-1})DH \leq (\frac{2}{n} + \frac{2}{n-2})LB$,

lo cual implica el siguiente teorema:

Teorema: *Para el TTP-2 con n equipos tal que* $n \equiv 0(4)$*, el algoritmo anterior corre en tiempo* $O(n^3)$ *y encuentra un schedule factible tal que la distancia viajada es a lo sumo* $1 + \frac{2}{n-2} + \frac{2}{n}$ *veces la distancia a viajar óptima.*

4.3.5 Búsqueda local haciendo Swapping

También se podría aplicar técnicas de búsqueda local más simples a este *schedule*. Estas técnicas podrían no mejorar la relación de aproximación, en teoría. De todas formas, en la práctica, para la mayoría de las instancias de referencia, podrían aún mejorar algo los resultados, alrededor de un 1%. Se podría usar:

- Hacer *swap* a dos pares de equipos en el matching M_H.
- Hacer *swap* a cualquier par de equipos.

4.4 El Traveling Tournament Problem de Múltiples Rondas Balanceado

R Hoshino y K Kawarabayashi [14] proponen un TTP de múltiples rondas balanceado (mb-TTP), inspirado por la estructura de la liga profesional de baseball de Japón, donde la cantidad de equipos n es 6, que juegan 120 partidos interliga en $r = 8$ rondas, sujeto a varias retricciones que aseguran el balance en la competitividad. Estas restricciones adicionales habilitan la reformulación del mb-TTP de 2k rondas como un problema de caminos mínimos en un grafo dirigido, para todos los $k \geq 1$. Se aplica un algoritmo teórico a la Liga Central Profesional de Baseball de Nippon, creando un *schedule* de distancia óptima con un total de distancia viajada de 57836 kilómetros, un 26.8% de reducción comparado con los 79067 kilómetros viajados por estos seis equipos durante la temporada regular del 2010.

De todas maneras, el TTP no es un contexto adecuado para el baseball, ya que los equipos juegan unos contra otros múltiples veces durante la temporada.

Lo que es más, ciertas ligas deportivas imponen restricciones de scheduling que van más allá del requerimiento de factibilidad del TTP. Por eso, se propuso una generalización del TTP: el mb-TTP.

Como el contexto es el baseball, se usan *sets* en vez de *días* o *fechas* para referirse a la longitud del torneo. A diferencia de otros deportes, donde un equipo visita otras ciudades para jugar un sólo partido, en las ligas de baseball, un equipo suele jugar varios partidos consecutivos de visitante contra un mismo equipo. Un set es un número fijo de partidos que se juegan en días (fechas) consecutivas. A continuación, se redefine el TTP para obtener el *schedule* de $2(n-1)$ sets. En el TTP, un *schedule* double round robin debe satisfacer las siguientes condiciones:

a) Cada par de equipos debe jugar entre sí dos sets, uno de local y otro de visitante.

b) Ningún equipo puede tener un break (local o visitante) que dure más de tres sets.

c) Un equipo no puede jugar contra el mismo oponente dos sets seguidos.

Para el caso $n = 6$, cada equipo juega cinco sets de local y cinco sets de visitante. Dado que un break de local es a lo sumo de tres sets, es más fácil ver que cada equipo debe realizar al menos siete *road trips*. Luego, en cualquier solución factible de TTP, se requieren al menos $7 \times 6 = 42$ *road trips* totales para los 6 equipos. En este caso, la cota inferior que se conoce es 43. Intuitivamente, la solución del TTP que minimiza la función objetivo de la distancia total viajada tendrá alrededor de 43 *road trips*.

Se define un *bloque* como una solución factible del TTP, es decir, un torneo que dura $2(n-1)$ sets. Se dice que un bloque consiste de dos rondas, siendo la primera ronda los primeros $n-1$ sets, y la segunda los restantes $n-1$ sets. En esta extensión de multi rondas, un torneo tendrá k bloques, lo cual equivale a tener $2k$ rondas o $2k(n-1)$ sets.

Por lo general, se dedica mucha más investigación al TTP donde no hay más de dos rondas. Estos se debe a la complejidad computacional de analizar un torneo que dure más de $2(n-1)$ días. Además de la dicha extensión a múltiples rondas, se consideran también dos restricciones de "balance" inspiradas en la estructura propia de la liga profesional de baseball de Japón.

Primero, el *schedule* de la NPB requiere que cada par de equipos se enfrente en exactamente un set durante la ronda de $n-1$ sets, lo cual es más fuerte que la

condicion (a). Notar que esta condición es similar pero no idéntica a la condición espejada de las ligas latinoamericanas de fútbol: si un equipo i recibe al equipo j en el set s (donde $1 \leq s \leq n-1$), luego el equipo j recibe al equipo i en el set $s+n-1$.

En segunda instancia, el *schedule* de la NPB requiere un balance en el número de sets de visitante y local que cada equipo juega en cualquier punto de la temporada. Más formalmente, para cada par ordenado (i, s) con $1 \leq i \leq n$ y $1 \leq s \leq 2k(n-1)$, se define $H_{i,s}$ y $R_{i,s}$ como el número de sets, de visitante y local, jugados por el equipo i dentro de los primeros s sets. Por definición, $|H_{i,s} - R_{i,s}| = s$. Aquí, se requiere que $|H_{i,s} - R_{i,s}| \leq 2$ para todos los pares (i, s). Por ejemplo, bajo este requerimiento, un equipo no puede empezar o terminar una temporada con tres sets consecutivos de local, asegurándose que ningún equipo gane cierta ventaja deportiva en un punto clave de la temporada.

Luego, obtenemos dos condiciones más:

d) Cada par de equipos debe enfrentarse exactamente una vez por ronda, con sus partidos en las rondas $2t-1$ y $2t$ con localías invertidas (para todo $1 \leq t \leq k$).

e) $|H_{i,s} - R_{i,s}| \leq 2$ para todo (i, s) con $1 \leq i \leq n$ y $1 \leq s \leq 2k(n-1)$.

Aquí, se presenta un algoritmo que resuelve el 2k-round mb-TTP que satisfaga las condiciones $(a) - (e)$. Para aplicar dicho algoritmo a la liga central NPB, se fija $k = 4$ y $n = 6$, ya que cada uno de los seis equipos juega $2k(n-1) = 40$ sets de tres partidos interliga en $r = 2k = 8$ rondas.

La condición (d) habilita el uso de la teoría de matchings perfectos, ya que cada ronda de cinco sets representa una 1-factorización del grafo completo K_6. Y, como se desmostrará más adelante, la condición (e) permite definir una simple "matriz de concatenación" con sólo cuatro columnas para verificar rápidamente las condiciones (b) y (c) cada vez de dos bloques de diez sets estén concatenados. Luego, sumando estas dos restricciones, el análisis se simplifica y hace que el algoritmo sea computacionalmente factible para el caso $n = 6$.

Para presentar el algoritmo, se crea un nodo fuente y un nodo sumidero, y los unimos a numerosos vértices en un grafo cuyas aristas pesadas representan los posibles bloques que pueden aparecer en un *schedule* óptimo. Luego, se aplica el algoritmo de Dijkstra para encontrar el camino de menor peso entre la fuente y el sumidero, que es un algoritmo de búsqueda de tiempo $O(|V|log|V| + |E|)$, que puede ser aplicado a cualquier grafo o digrafo con aristas de peso no negativo.

4.4.1 Camino más corto para el mb-TTP

El mb-TTP puede ser formulado como un problema de programación entera con $2k(n-1)$ intervalos de tiempo con cada uno representando un set. Pero dicho problema tendría k veces la cantidad de variables que tiene la formulación del problema original para el 2-round TTP, e incluso para el caso con $n = 6$, puede no ser computacionalmente factible para un k grande. Como resultado, se propone una reforumación del problema de optimización a través de un problema de caminos mínimos en un digrafo. A pesar de ser una metodología computacionalmente trabajosa para k pequeño, resulta ser muy útil para grandes valores de k.

Para resolver el mb-TTP, primero computamos el conjunto de bloques que pueden aparecer en un torneo de distancia óptima. Luego, empleamos la matriz de concatenación para chequear si los dos bloques precomputados pueden unirse para formar un *schedule* de múltiples bloques, sin violar las condiciones (b) y (c). Como se explicará más adelante, para determinar si los bloques B_1 y B_2 pueden ser concatenados, es suficiente chequear sólamente las últimas dos columnas de B_1 y las primeras dos de B_2.

Por definición, un bloque es un torneo que satisface la condición del mb-TTP, con cada uno de los equipos jugando $2(n-1)$ sets de partidos. Cada columna de un bloque representa un set que consiste en $\frac{n}{2}$ *matches* (emparejamientos), siendo que cada match especifica los dos equipos que disputan el partido y también la localía. Luego, un match identifica el equipo local y el visitante, no sólo el equipo oponente.

Por cada columna en un bloque, hay $\binom{n}{\frac{n}{2}}$ formas de elegir los equipos locales. También, hay $\binom{n}{\frac{n}{2}}.\frac{n}{2}!$ formas de especificar los matches en cualquier columna, pues hay $\frac{n}{2}!$ formas de mapear cualquier elección de los $\frac{n}{2}$ equipos locales contra los aún no seleccionados $\frac{n}{2}$ equipos visitantes para decidir el conjunto de $\frac{n}{2}$ matches. Luego, hay m= $\binom{n}{\frac{n}{2}}^2.\frac{n}{2}!$ formas diferentes de especificar los equipos locales de una columna y los matches de otra columna. Para $n = 6$, tenemos $m = \binom{6}{2}^2 \times 3! = 2400$.

Hay m formas de elegir las primeras dos columnas de un bloque, como se describió anteriormente, con la primera columna mostrando los matches y, la segunda, mostrando los equipos locales. Luego, se usa cualquier método, por ejemplo, el ordenamiento léxicografico, para indexar estas m opciones con los enteros del 1 al m. Por simetría, hay m formas diferentes de especificar las últimas dos columnas de un bloque, con la última oclumna mostrando los matches y la anteúltima columna mostrando los equipos locales. Luego, usamos el mismo esquema para

indexar estas m opciones. Para evitar confusiones, escribimos los equipos locales en forma binaria, con 1 representando un partido de local y 0 representando un partido de visitante.

Para cada par (u_1, u_2), con $| \leq u_1, u_2 \leq m$, definimos C_{u_1,u_2} como la matriz de concatenación de $n \times 4$ donde las primeras dos columnas muestran los equipos locales y los matches con el índice u_2, y las siguientes dos columnas muestran los matches y los equipos locales con índice u_1.

A continuación, se explica el rol de m y C_{u_1,u_2} en la construcción de nuestro grafo dirigido. Sea G un grafo que consiste en un vértice fuente v_{start}, un vértice sumidero v_{end} y vértices $x_{t,u}$ y $y_{t,u}$ definidos para cada $1 \leq t \leq k$ y $1 \leq u \leq m$.

La descripción de cómo se conectan las aristas es la siguiente (se denota $v_1 \to v_2$ a la arista de v_1 a v_2):

1. Para cada $1 \leq u \leq m$, se agrega la arista $v_{start} \to x_{1,u}$.
2. Para cada $1 \leq u \leq m$, se agrega la arista $y_{k,u} \to v_{end}$.
3. Para cada $1 \leq t \leq k$ y $1 \leq u_1, u_2 \leq m$, se agrega la arista $x_{1,u_1} \to y_{t,u_2}$ si y sólo si existe un bloque factible en el cual la primeras dos columnas tienen índice u_1 y las últimas dos columnas tienen índice u_2.
4. Para cada $1 \leq t \leq k-1$ y $1 \leq u_1, u_2 \leq m$, se agrega la arista $y_{t,u_2} \to x_{t+1,u_1}$, si y sólo si la matriz de concatenación C_{u_1,u_2} no tiene ninguna fila con cuatro sets de local, ninguna fila con cuatro sets de visitante y ninguna fila con el mismo oponente apareciendo en las columnas 2 y 3.

El siguiente teorema muestra que el mb-TTP puede ser reformulado en un contexto de teoría de grafos, para todo $k \geq 1$.

Teorema 1: *Toda solución factible del mb-TTP puede ser descripta por un camino de v_{start} a v_{end} en un grafo G. Recíprocamente, cualquier camino de v_{start} a v_{end} en G corresponde a una solución factible del mb-TTP.*

Demostración: por la definición de G, cualquier camino desde v_{start} hasta v_{end} tiene longitud $2k + 1$, y es de la forma $P = v_{start} \to x_{1,p_1} \to y_{1,q_1} \to x_{2,p_2} \to y_{2,q_2} \to ... \to x_{k,p_k} \to y_{k,q_k} \to v_{end}$.

Primero, mostraremos que P corresponde a un *schedule* que es una solución factible del mb-TTP. Para cada $1 \leq t \leq k$, sea B_t cualquier bloque para el cual las primeras dos columnas tienen índice p_t y las últimas dos columnas tienen índice q_t. Por la parte (c) de nuestra construcción, ya que $x_{t,p_t} \rightarrow y_{t,q_t}$ es una arista de G, tal bloque B_t debe necesariamente existir. Como cada B_t es un bloque (factible), todas las resetricciones de balance del mb-TTP se cumplen en ese bloque: cada par de equipos se enfrenta exactamente una vez por ronda con un partido en cada estadio, ningún equipo tiene un break de local o visitante que dure más de tres sets, un equipo no juega contra el mismo oponente en sets consecutivos y $|H_{i,s} - R_{i,s}| \leq 2$ para todo (i, s) con $1 \leq i \leq n$ y $1 \leq s \leq 2(n-1)$.

Decimos que $B_1, B_2, ..., B_k$, la concatenación de estos k bloques, es una solución factible del mb-TTP. Claramente cada equipo juega $2k(n-1)$ sets de partidos, cada par de equipos juega una vez por ronda con su match en las rondas $2t-1$ y $2t$ en diferentes estadios (para todo $1 \leq t \leq k$). Como cada equipo juega $n-1$ sets de local y $n-1$ sets de visitante en cada bloque, tenemos $H_{i,2t(n-1)} = R_{i,2t(n-1)}$ para cada $1 \leq t \leq k$. Esto implica que $|H_{i,s} - R_{i,s}| \leq 2$ para todo (i, s) con $1 \leq i \leq n$ y $1 \leq s \leq 2k(n-1)$, pues esta función distinta $|H_{i,s} - R_{i,s}|$ se resetea a 0 al final de cada bloque. Entonces, las condiciones (d) y (e) se satisfacen.

Para completar la suposición, debemos justificar que las condiciones (b) y (c) no son violadas. Basado en lo dicho anteriormente, sólo se puede crear unaviolación en la concatenación de dos bloques, específicamente en una de las siguientes situaciones:

(1) La última columa del bloque B_t se empareja bien con la primera columna del bloque B_{t+1} en, al menos, una fila. Es decir, que existe un equipo que juega contra el mismo oponente en los sets $2t(n-1)$ y en $2t(n-1)+1$, así violando la condición (c).

(2) Alguna fila de B_t termina con dos sets de local (o de visitante) y la misma fila de B_{t+1} empieza con dos sets de local (o de visitante), así violando la condición (b) del set $2t(n-1)-1$ al $2t(n-1)+2$.

(3) Alguna fila de B_t termina con tres sets de local (visitante) y la misma fila de B_{t+1} empieza con tres sets de local (visitante), así violando la condición (b) del set $2t(n-1)-2$ al $2t(n-1)+1$.

(4) Alguna fila de B_t termina con un set de local (visitante) y la misma fila de B_{t+1} empieza con tres sets de local (visitante), así violando la condición (b) del set $2t(n-1)$ al $2t(n-1)+3$.

Consideremos los equipos que juegan de local en la anteúltima columna de B_t, los matches en la última columna de B_t, los matches en la primera columna de B_{t+1}, y los equipos que juegan de local en la segunda columna de B_{t+1}. Estas cuatro columnas, que representan los sets desde el la condición (b) del set $2t(n-1)-1$ al $2t(n-1)+2$, son precisamente las cuatro columnas de la matriz de concatenación $C_{q_t,p_{t+1}}$. Dado que el camino P incluye a la arista $y_{t,q_t} \to x_{t+1,p_{t+1}}$, la matriz $C_{q_t,p_{t+1}}$ no tiene ninguna fila con cuatro sets de local, ninguna fila con cuatro sets de vissitante ni ninguna fila con el mismo oponente que aparezca en Columna 2 y Columna 3. Luego, ningún equipo tiene un break de cuatro sets de local o un *road trip* desde el set $2t(n-1)-1$ al $2t(n-1)+2$, y ningún equipo juega contra el mismo oponente en los sets $2t(n-1)$ y $2t(n-1)+1$. Esto prueba que las situaciones (1) y (2) no pueden ocurrir.

Como se mencionó antes, $H_{i,2t(n-1)} = R_{i,2t(n-1)}$, para todo t. Si la fila i de B_t termina con tres sets consecutivos de local o de visitante, entonces $|H_{i,2t(n-1)-3} - R_{i,2t(n-1)-3}| = 3 > 2$, lo cual es una contradicción. De la misma manera, si la fila i de B_{t+1} empieza con tres sets consecutivos de local o visitante, entonces $|H_{i,2t(n-1)+3} - R_{i,2t(n-1)+3}| = 3 > 2$, una contradicción. Esto prueba que ni la situación (3) ni la (4) pueden ocurrir.

Hemos mostrado que la concatenación de estos k bloques $(B_1, ..., B_k)$ es una solución factible del mb-TTP. Para concluir la prueba, veamos la vuelta.

Notemos que cualquier solución factible del mb-TTP puede ser particionada en k bloques no superpuestos. Sea B_t el $t^{\text{ésimo}}$ bloque de esta solución factible. Para este bloque B_t, sea p_t el índice de las primeras dos columnas y q_t el índice de las últimas dos columnas. Como B_t es un bloque (factible), cumple que cada $x_{t,p_t} \to y_{t,q_t}$ es una arista de G. Como dos bloques emparejados pueden ser concatenados sin violar ninguna restricción del mb-TTP, tenemos $y_{t,q_t} \to x_{t+1,p_{t+1}}$ en una arista de G para todo $1 \leq t \leq k-1$. Concluimos que $P = v_{start} \to x_{1,p_1} \to y_{1,q_1} \to x_{2,p_2} \to y_{2,q_2} \to ... \to x_{k,p_k} \to y_{k,q_k} \to v_{end}$ es un camino de G que conecta v_{start} con v_{end}. ■

Habiendo construido nuestro digrafo, ahora asignamos un peso a cada arista usando la matriz D de distancias, tal que el camino mínimo de v_{start} a v_{end} corresponda a la solución deseada de mb-TTP que minimice el total de la distancia viajada por los n equipos.

Para cualquier bloque, definimos su *distancia interna* como la distancia total viajada por los n equipos dentro de ese bloque, es decir, empezando desde el set 1 y terminando en el set $2(n-1)$. Se usará esta definición en la siguiente parte (c).

Ahora, se pueden asignar los pesos a G.

(a) Para cada $1 \leq u \leq m$, el peso de $v_{start} \to x_{1,u}$ es la distancia viajada por los $\frac{n}{2}$ equipos que hacen el viaje desde su ciudad hasta el estadio de su oponente del set 1.

(b) Para cada $1 \leq u \leq m$, el peso de $y_{k,u} \to v_{end}$ es la distancia viajada por los $\frac{n}{2}$ equipos que viajan del estadio de su oponente en el set $2k(n-1)$ de vuelta a su ciudad.

(c) Para cada $1 \leq t \leq k$ y para cada $1 \leq u_1, u_2 \leq m$, el peso de $x_{1,u_1} \to y_{t,u_2}$ es la mínima distancia interna de un bloque, seleccionado entre todos los bloques de los cuales las primeras dos columnas tienen índice u_1 y las últimas dos columnas tienen índice u_2.

(d) Para cada $1 \leq t \leq k-1$ y para cada $1 \leq u_1, u_2 \leq m$, el peso de la arista $y_{t,u_2} \to x_{t+1,u_1}$, es la distancia viajada por todos los equipos que viajan desde la ciudad de su match en el set $2t(n-1)$ a su match en el set $2t(n-1)+1$ donde las últimas dos columnas del $t^{\text{ésimo}}$ bloque tiene índice u_2 y las primeras dos columnas del $(t+1)^{\text{ésimo}}$ bloque tiene índice u_1

Para ilustrar (d), consideremos el *schedule* de dos bloques producido por la concatenación de dos copias de los primeros diez sets del *schedule* del 2010 de la Liga Central. Luego, esta es una solución factible del mb-TTP para $k = 2$, con el camino $P = v_{start} \to x_{1,p} \to y_{1,q} \to x_{2,p} \to y_{2,q} \to v_{end}$ de peso total $1010 + 15895 + 1707 + 15895 + 1697 = 3604$. El término 1707, que representa el peso total de la arista $y_{1,q} \to x_{2,p}$, es la distancia viajada por los equipos desde sus matches en el set 10 hasta sus matches en el set 11. La información de esas distancias nos la proporciona la matriz de distancias D.

Por esta construcción, se produce un digrafo ponderado. Para el pas (c), suponemos que existen dos bloques B y B' para los cuales las primeras dos columnas tienen índice u_1 y las últimas dos columnas tienen índice u_2. Si la distancia interna de B es menor que la distancia interna de B', entonces el bloque B' no puede ser un bloque en una solución óptima, ya que sólo podemos reemplazar B' por B para crear una solución factible con un valor de la función objetivo más chico. Esta observación trivial, basada en el Principio de Optimalidad de Bellman, permite asitnar la distancia interna mínima como el peso de la arista $x_{t,u_1} \to y_{t,u_2}$, para todo $1 \leq u_1, u_2 \leq m$. Como resultado, tenemos el digrafo G con $2mk + 2$ vértices y a lo sumo $2m + (2k-1)m^2$ aristas, con pesos únicos para cada una.

Combinando con el teorema anterior, se acaba de demostrar lo siguiente:

Teorema 2: *Sea* $P = v_{start} \to x_{1,p_1} \to y_{1,q_1} \to x_{2,p_2} \to y_{2,q_2} \to ... \to x_{k,p_k} \to y_{k,q_k} \to v_{end}$ *el camino mínimo de G desde* v_{start} *hasta* v_{end}*, es decir, el camino que minimiza el peso total. Para cada* $1 \leq t \leq k$*, sea* B_t *el bloque de menor distancia internaelegido entre todos los bloques en los cuales las primeras dos columnas tienen índice* p_t *y las últimas dos columnas tienen índice* q_t*. Luego, el schedule de múltibloques* $S = B_1, B_2, ..., B_k$ *creado por la concatenación de estos k bloques, es una solución óptima del mb-TTP*

4.4.2 Construcción del Digrafo

Se mostró que el mb-TTP es isomorfo a encontrar el camino mínimo de menor peso en el grafo dirigido G. Se construye G para para los 6 equipos de la Liga Central de la NPB, usando la matriz de distancias dada D.

Primero, notar que para los pasos (c) y (d), estas dos construcciones son independientes de t. Entonces, para construir las aristas $x_{t,u_1} \to y_{t,u_2}$, es suficiente determinar todas las aristas desde x_{t,u_1} hasta y_{t,u_2} con sus correspondientes pesos y replicar eso k veces. Para construir las aristas $y_{t,u_2} \to x_{t+1,u_1}$ es suficiente determinar todas las aristas desde y_{t,u_2} hasta y_{2,u_1} con los correspondientes pesos y replicar eso k-1 veces. Independientemente del k, sólo necesitamos computar los pasos (c) y (d) una vez, y luego guardamos toda la información necesaria en matrices de $m \times m$, las cuales serán computadas como matrices de dimensiones 2400×2400 para el caso $n = 6$.

Para realizar el paso (c) de la construcción, se debe asegurar que el peso de cada arista $x_{1,u_1} \to y_{1,u_2}$ sea la mínima distancia interna de un bloque en el cual las primeras dos columnas tienen índice u_1 y las últimas dos columnas tienen índice u_2. Para poder lograr eso, se adopta una heurística común que consta de tres pasos para resolver el 2 round TTP (Rasmussen y Trick 2007 [15]). La fase uno genera los sets de un double round robin home-away pattern (HAP) de la forma de una matriz de $n \times 2(n-1)$. La fase dos convierte esos sets HAP a esquemas que asignan los partidos a intervalos de tiempo. Y la fase tres convierte esos esquemas en *schedules* factibles (i.e., bloques) asignando a cada equipo una fila única de la

matriz. Una vez que se enumeraron todos los posibles bloques, se puede asignar el peso apropiado a cada arista $x_{1,u_1} \to y_{1,u_2}$ aplicando la matriz de distancias D.

Para transformar un esquema en un bloque, simplemente se mapea los seis equipos de la Liga Central en cualquiera de las 6! permutaciones de $1, 2, 3, 4, 5, 6$. Notar que cualquier esquema puede ser transformado en un bloque, pero un set HAP no necesariamente produce un esquema factible. Por ejemplo, considerar un set HAP con dos filas idénticas i y j. Luego, este set HAP no producirá un esquema factible, dado que los equipos i y j no pueden jugar encontra pues no hay un intervalo de tiempo donde un de los dos equipos juegue de local mientras el otro juega de visitante. Usando las factorizaciones-1, se puede mostrar que hay 627944 sets HAP posibles, que generan 169728 esquemas factibles.

De esos 169728 esquemas, se calcula la distancia interna de los $169728 \times 6!$ bloques que pueden ser generados y determinan los índices de sus primeras y últimas dos columnas. Por esto, determinamos que la matriz M de 2400×2400, donde $M_{[u_1,u_2]}$ es la distancia interna mínima de un bloque, elegido entre todos los bloques en los cuales las primeras dos columnas tienen índice u_1 y las últimas dos columnas tienen índice u_2. Se determina que 2618520 de los $2400^2 = 5760000$ posibles lugares de la matriz M están definidos, todos cuyos valores estén en el rango $[13075, 22189]$.

Para los $2400^2 - 2618520 = 3141480$ casos restantes, no existe un bloque factible con índices u_1 y u_2, entonces definimos $M_{[u_1,u_2]} := 10^5$ en estos casos, para asegurar que la matriz M esté bien definida. En un algoritmo de caminos mínimos, cuando dos vértices no son adyacentes, se puede crear una arista falsa que tenga peso masivo, así será más pesada que el resto de las aristas existentes del grafo. Como resultado, toda la información de las aristas y los pesos de $x_{1,u_1} \to y_{1,u_2}$ pueden guardados en esta matriz M, y cuando se aplique el algoritmo de Dijkstra para encontrar el camino que minimice el peso total, una arista de peso tan grande no va a aparecer como parte del output del algoritmo.

Finalmente, procedemos con el paso (4) de nuestra construcción. Definimos la matriz N de 2400×2400 que va a guardar los pesos de todas las aristas $y_{1,u_2} \to x_{2,u_1}$. Por definición, $y_{1,u_2} \to x_{2,u_1}$ es una arista en G si y sólo si la matriz de concatenación C_{u_2,u_1} no tiene ninguna fila con cuatro sets locales, ninguna fila con cuatro sets visitantes ni ninguna fila con el mismo oponente en las columnas 2 y 3. Encontramos que 1486320 de los $2400^2 = 576000$ elecciones de C_{u_2,u_1} satisfacen los requerimientos de que $y_{1,u_2} \to x_{2,u_1}$ sea una arista de G. En todos estos casos, la arista pesada $N_{[u_2,u_1]}$ está determinada por C_{u_2,u_1} calculando las distancias viajadas por todos los equipos que hacen un viaje desde la ciudad de su match en la

columna 2 hasta su match de la columna 3. Así, se ecuentra que 1468320 de los valores de $N_{[u_2,u_1]}$ viven en el rango $[79, 3394]$. En los otros $2400^2 - 1486320 = 4273680$ casos, se fija $N_{[u_2,u_1]} := 10^5$, así nos definimos que la matriz esté bien definida.

Claramente, los pasos (a) y (b) en nuestra construcción agregan cada uno 2400 aristas a G. Por lo dicho antes, el paso (c) agrega 2618520k aristas y el (d) agrega 1486320(k-1). Entonces, el grafo G consiste en $2mk + 2 = 4800k + 2$ vértices y $2400 + 2400 + 2618520k + 1486320(k-1) = 4104840k - 1481520$ aristas. Ahora, se aplica el algoritmo de caminos mínimos de Dijkstra para obtener el 2k-round *schedule* de k-bloques que minimiza la distancia total.

El algoritmo de Dijkstra construye un árbol de camino mínimo desde el vértice inicial hasta todos los demás vértices en el grafo. Por cada vértice v en G, sea $w(v)$ el peso del camino mínimo desde v_{start} hasta ese vértice. Por definición, $w(v_{start}) = 0$. Para cada $1 \leq u \leq m$, $w(x_{1,u})$ es la distancia total viajada por los $\frac{n}{2}$ equipos que realizan el viaje desde su ciudad hasta la ciudad del oponente en el set 1.

Para cada $1 \leq u \leq m$ y $1 \leq t \leq k$, tenemos

$w(y_{t,u}) = min_{1 \leq u* \leq m}\{w(x_{t,u*}) + M_{[u*,u]}\}$

Y para cada $1 \leq u \leq m$ y $1 \leq t \leq k-1$, tenemos

$w(x_{t+1,u}) = min_{1 \leq u* \leq m}\{w(y_{t,u*}) + N_{[u*,u]}\}$.

El último paso de esta doble recursión genera $w(y_{k,u})$, el peso del camino mínimo desde v_{start} hasta cada $y_{k,u}$. Para cada u, se define $last(u)$ como el total de la distancia viajada por los $\frac{n}{2}$ equipos que realizan el viaje desde la ciudad de su oponente en el set $2k(n-1)$ hasta su propia ciudad. Por simetría, es claro que $last(u) = w(x1, u)$. Luego, tenemos

$w(v_{end}) = min_{1 \leq u* \leq m}\{w(y_{k,u*}) + w(x_{1,u*})\}$.

El valor de $w(v_{end})$ es la distancia de la solución óptima del k-bloque mb-TTP, correspondiente a algun camino $P = v_{start} \to x_{1,p_1} \to y_{1,q_1} \to x_{2,p_2} \to y_{2,q_2} \to ... \to x_{k,p_k} \to y_{k,q_k} \to v_{end}$. Para cada $1 \leq t \leq k$, sea B_t el bloque de distancia interior mínima elegida entre todos los bloques que tienen las primeras dos columnas con índice p_t y las últimas dos columnas con índice q_t. Por el Teorema 2, la el *schedule* deseado de distancia mínima es la concatenación de $B_1, B_2, ..., B_k$.

4.4.3 Aplicación a la Liga NPB japonesa

Se aplicó el algoritmo de caminos mínimos para optimizar el scheduling de la liga profesional de baseball de Japón. La NPB consiste en una Liga Central de seis equipos y en una Liga del Pacífico de seis equipos. Todos los equipos juegan 144 partidos durante la temporada, con 120 partidos dentro de la liga y 24 partidos de otra liga. Específicamente, un equipo de la NPB juega 12 partidos de local y 12 partidos de visitante contra cada uno de los otros equipos en su propia liga, a eso se le suma dos partidos de local y dos de visitante en contra de los seis equipos de la otra liga. Los 24 partidos fuera de la liga se juegan durante cinco semanas a mitad de Mayo, cerca del comienzo de la temporada.

El algoritmo propuesto aquí para el mb-TTP sólo se aplica a los 120 partidos dentro de la liga, los restantes 24 partidos constituyen un problema de optimización diferente.

Durante la temporada de la NPB del 2010, todo equipo de la Liga Central realizó al menos 33 viajes. Bajo el *schedule* óptimo al que se llegó, ningun equipo debió realizar más de 29 viajes. También, este *schedule* alcanzó un 26.8% de reducción de la distancia total viajada comparando con *schedules* preexistentes, además de una reducción del 14.6% en el total de los viajes realizados.

5 Scheduling de Torneos de Básquet

5.1 Introducción

Resolver scheduling de torneos de básquet con métodos exactos representa una gran dificultad dada la enorme cantidad de variables y valores que se deben considerar. Luego, para este caso, se empezaron a usar algoritmos heurísticos.

Una de las primeras investigaciones sobre el tema en la modernidad se publicó en 1976. Allí fue cuando Campbell y Chen [5] desarrollaron un algoritmo para minimizar las distancias viajadas por los equipos en una liga de básquet de una universidad. Este algoritmo tenía dos fases. La primera era encontrar un *schedule* con una distancia mínima para viajar para cada equipo. Para eso, los equipos son divididos en pares basados en las distancias. El patrón de viaje óptimo para un equipo es viajar a todo el resto de los pares en un *road trip* antes de volver a casa, y usar un *road trip* de una ronda para visitar al equipo que esta emparejado con éste. En la segunda fase, todas las demás restricciones son aplicadas para crear una solución factible. En este caso, se reportó un descenso significativo en la distancia viajada por cada equipo al usar este algoritmo.

Inspirándose en ellos, Bean y Birge [1] trataron de atacar el problema de la NBA con un algoritmo similar, del cual ya se habló en la **sección 2**. Además de ese algoritmo, se construyeron otros algoritmos heurísticos para crear *schedules* factibles. Por ejemplo, Froncek [12] usó un algoritmo basado en teoría de grafos para crear un *schedule* para la liga nacional de básquet de República Checa.

Con el desarrollo de la tecnología, resolver este problema con un método exacto se volvió posible. Nemhauser y Trick [19] enfrentaron un problema al crear el *schedule* para los partidos de básquet en la Conferencia de la Costa Atlántica (ACC) del torneo universitario de Estados Unidos. Se hizo en tres pasos. Primero, se encontró los patrones de juego y los conjuntos de patrones. Se pudieron generar los 38 posibles patrones de juego para un equipo. Todas las restricciones relacionadas con el patrón de juego se consideraron al generar los patrones, por ejemplo, la cantidad de partidos de local y los partidos jugados durante un fin de semana. Luego, se generaron los 17 conjuntos de patrones. Y, aquí, también se consideraron todas las restricciones relacionadas al *time slot*. El segundo paso, fue asignar los equipos a los conjuntos de patrones para generar el fixture. Se tardo 10 minutos en generar 826 fixtures factibles. Y el último paso, fue asignar el equipo a los fixtures, lo que lleva la mayor cantidad de tiempo. Luego, se tardó 24 horas en generar 17 *schedules* factibles.

Si nuestro objetivo fuera encontrar una solución factible, más que un valor óptimo, luego, *constraint programming* es más fuerte comparado con la programación entera. Henz [13] mostró eso al resolver el problema de Trick en mucho menos tiempo de cómputo. Produjo resultados similares, pero redujo el tiempo total de cómputo de 24 horas a 1 minuto. El problema se descompuso en tres fases, todas resueltas con *constraint programming*.

Como Trick, muchos otros investigadores usaron programación entera para resolver problemas de ligas de básquet. Voorhis [20] desarrolló un modelo de programación entera para resolver el problema de scheduling de la liga de básquet para dos conferencias de equipos de universidades.

A pesar de que los métodos exactos son prometedores, los algoritmos heurísticos son necesarios en muchos casos. Wright [26] usó una metaheurística para crear el *schedule* para la liga de básquet de Nueva Zelanda. En vez de poner todos los requisitos como restricciones, el autor usó una función objetivo para modelar requerimientos subjetivos. Se usó un algoritmo de recosido simulado para resolver el problema con la menor penalidad posible.

5.2 Scheduling de la Liga de Básquet Profesional de Argentina: una Variación del Relaxed Traveling Tournament Problem

En este caso [6], los partidos se juegan cualquier día de la semana y los partidos de visitante se programan en *road trips* de uno a cuatro partidos consecutivos, para así evitar que los equipos deban regresar a su ciudad local antes de jugar cada partido de visitante. El objetivo principal es reducir la distancia total que viajan los equipos durante todo el torneo comparado con los formatos de las temporadas anteriores mediante el uso de *road trips* propuestas por los propios equipos. Es por eso que se utilizó una variación del Traveling Tournament Problem, creando un modelo que se divide en dos etapas consecutivas. La primera, define el orden de los partidos y su localía; y, la segunda, asigna los días del calendario en los que los partidos se disputarán. Ambas fases usan modelos de programación entera que consideran restricciones que reflejan los requerimientos de la Asociación de clubes de Básquetbol Argentina .

5.2.1 Historia del los Torneos de Básquet Profesional en Argentina

En 2013-2014, la primera división tenía 16 equipos y una temporada regular que consistía en dos partes: una regional y otra nacional. En la etapa regional, los equipos se dividían en dos zonas de 8 equipos (Norte y Sur) en las que cada equipo se enfrentaba a todos los de su zona una vez de local y una vez de visitante. Luego, seguía la etapa nacional que usaba el mismo formato para los 16 equipos como un grupo en sí. En total, cada equipo jugaba 44 partidos en la fase regular de la temporada.

Los partidos se jugaban los viernes y domingos usando un sistema de pares en el cual los equipos se dividían en grupos de dos. Cada fin de semana, un par $A = (A_1, A_2)$ visitaba un par $B = (B_1, B_2)$. El viernes, A_1 se enfrentaría a B_2, mientras que A_2 se enfrentaría a B_1. Luego, el domingo, se invertía dicha combinación. Un fin de semana de la etapa regional y otro de la nacional, se dedicarían exclusivamente a partidos *intra pares*, en los cuales cada pareja se enfrentaría entre sí dos veces. Para que este sistema tuviera sentido, se procuraba que los equipos de cada pareja estuvieran localizados relativamente cerca unos de otros; así, el par visitante tendría que recorrer menos distancia entre el viernes y el domingo.

Uno de los aspectos centrales de este formato de *schedule* aplicado por estos modelos es el uso de *road trips* similar al de la NBA de Estados Unidos. El modelo se divide en dos etapas. La primera, es un modelo que define el schedule con el orden de los partidos teniendo en cuenta los *road trips* posibles (los preferidos por los equipos y los razonables) con la idea de reducir las distancias viajadas. Esta fase es una nueva variante del *Time Relaxed Traveling Tournament Problem* a la cual se la denomina *Trip Preferences-Time Relaxed Traveling Tournament Problem*. La segunda etapa es un modelo que le asigna días exactos a cada partido definido en el anterior. Ambos incluyen una serie de restricciones que reflejan las peticiones de la Asociacion de Clubes de Básquetbol Argentina, las televisoras y los equipos.

5.2.2 Primera Etapa del Modelo: programación de los road trips y el orden de los partidos

Este primer modelo toma como principal input los posibles *road trips* para cada equipo. No define los días exactos en los que se disputarán los partidos, pero considera que como máximo se programarán tres partidos por semana, los cuales, para esta etapa, se limitan a lunes, miércoles y viernes (en la segunda etapa, el modelo va a asignar los partidos a cualquier día de la semana). De todas maneras, en cada semana, un equipo no necesariamente va a jugar las tres fechas, ya que hay más posibles días de partido que fechas por equipo en una temporada. La idea es

que cada equipo juegue en promedio dos partidos por semana, con lo cual tendría en promedio una fecha de descanso por semana, lo que llamamos *bye*. Teniendo en cuenta todo lo dicho anteriormente, la idea de esta primera fase es maximizar la cantidad de partidos asignados a cada ronda haciéndola coincidir con las preferencias de los equipos para los *road trips*.

Input:

- conjunto E de equipos
- Para cada equipo $i \in E$, un conjunto $T(i)$ de *road trips* posibles. Cada *road trips* t es una sucesión de 1 a 4 equipos distintos de i. El conjunto de todos los *road trips* es $T = \cup_{i \in E} T(i)$. Además, para cada equipo i, $T(i) = PT(i) \cup RT(i)$ donde $PT(i)$ es el conjunto de *road trips preferidos* que el equipo eligió, y $RT(i)$ es el conjunto de *road trips razonables* sugeridos por la Asociación de Clubes.
- un total de n rondas. El conjunto de todas las rondas es $F = \{1, \ldots, n\}$.
- el conjunto de partidos que va a jugar de local y de visitante cada equipo $i \in E$ que deben ser programados en la temporada. La cantidad total de partidos que cada equipo disputará es estrictamente menor a n, lo cual implica que cada equipo tiene un conjunto de *byes* durante toda la temporada.
- un conjunto C de *cutoff dates*, el cual es predefinido por la Asociación de Clubes, y refleja la existencia de semanas de descanso incluidas en el calendario del torneo. Luego, un *road trip* debe completarse antes del comienzo de una de estas semanas de descanso. Definimos como *cutoff date* de cada semana al viernes anterior al comienzo de la semana de descanso.

El modelo contiene la variable binaria z_{tk} para cada *road trip* $t \in T$ y $k \in F$ donde $k \leq n - |t| + 1$ tal que $z_{tk} = 1$ si y sólo si el *road trip* t empieza con la ronda k. Adicionalmente, se define la variable x_{ijk} para cada par de equipos $i, j \in E$, $i \neq j$ y cada ronda $k \in F$ tal que $x_{ijk} = 1$ si y sólo si el equipo i juega de local contra el equipo j en la ronda k.

Luego, definimos formalmente el modelo:

La función objetivo busca maximizar la cantidad de partidos que se programan acorde a los *road trips* preferidos por los equipos:

$$\max \sum_{i \in E} \sum_{t \in PT(i)} \sum_{k \in F} |t| \, z_{tk}.$$

Las restricciónes son las siguientes:

1. Todos los partidos deben estar programados.

$$\sum_{k \in F} x_{ijk} = 1 \qquad \forall \, i, j \in E, i \neq j. \tag{1}$$

2. Cada equipo juega como máximo un partido por ronda. Notar que esta restricción implica que ningun equipo puede estar involucrado en más de un *road trips* al mismo tiempo.

$$\sum_{j \in E} \left(x_{ijk} + x_{jik} \right) \leq 1 \qquad \forall \, i \in E, \forall \, k \in F. \tag{2}$$

3. Los partidos que se juegan son elegidos del conjunto de *road trips* posibles.

$$x_{ijk} = \sum_{t \in T(j)} z_{t,(k - \mathrm{pos}(t,i))} \qquad \forall \, i, j \in E, i \neq j, \forall \, k \in F. \tag{3}$$

   En vistas de la definición que se dió de x_{ijk}, el valor de la variable es 1 si y sólo si al equipo j se le asigna un *road trip* que incluye un partido contra i en la ronda k. Dado un equipo $i \in E$ y un *road trip* $t \in T$ que incluye al equipo i, se define $\mathrm{pos}(t, i) \in \{0, \ldots, |t| - 1\}$ como la posición de i en la sucesión de partidos del *road trip* t.

4. Cada equipo juega al menos una vez de local en las cuatro rondas que siguen a un *road trip*.

$$\sum_{j \in E} \sum_{s=0}^{3} x_{ij,k+|t|+s} \quad \geq \quad z_{tk} \qquad \begin{array}{l} \forall \, i \in E, \forall \, t \in T(i), \\ \forall k \in F, k + |t| + 3 \leq n. \end{array} \tag{4}$$

   Esta restricción asegura que a ningún equipo se le asignen *road trips* consecutivos excesivamente largos.

5. Ningún *road trip* puede empezar antes o después de un *cutoff date*

$$z_{tk} = 0 \qquad \forall\, c \in C, \forall\, t \in T, \forall\, k \in F, k \leq c < k + |t| - 1. \tag{5}$$

6. Ningún *road trip* puede asignarse si no hay suficientes rondas restantes en la temporada para que éste se complete.

$$z_{tk} = 0 \qquad \forall\, t \in T, \forall\, k \in F, k > n - |t| + 1. \tag{6}$$

7. Cada equipo tiene un *bye* antes o después de cada *road trip* para darles un descanso a los jugadores, a menos que el *road trip* termine en la última ronda de la temporada.

$$\sum_{j \in E}(x_{ij,k+|t|} + x_{ji,k+|t|}) + \sum_{j \in E}(x_{ij,k-1} + x_{ji,k-1}) \quad \leq \quad 2 - z_{tk}$$
$$\forall\, i \in E, \forall\, t \in T(i), \forall\, k \in F, 1 < k \leq n - |t|. \tag{7}$$

8. Ningún equipo puede tener más de 3 *byes* seguidos, para evitar períodos extensos sin partidos.

$$\sum_{j \in E}\sum_{s=0}^{2} x_{ij,k+s} + x_{ji,k+s} \;\geq\; 1 \qquad \forall\, i \in E, \forall\, k \in F, k + 2 \leq n. \tag{8}$$

9. Un equipo debe tener al menos un partido de visitante cada $\ell + 1$ rondas.

$$\sum_{j \in E}\sum_{s=0}^{\ell} x_{ji,k+s} \;\geq\; 1 \qquad \forall\, i \in E, \forall\, k \in F, k + \ell - 1 \leq n. \tag{9}$$

   Esta restricción previene la "saturación" de fanáticos locales. Por lo general, se considera $\ell = 5$.

10. Cada equipo juega al menos un partido en las primeras tres rondas.

$$\sum_{j \in E}\sum_{k=1}^{3}(x_{ijk} + x_{jik}) \geq 1 \qquad \forall\, i \in E. \tag{10}$$

11. En la última ronda, todos los equipos deben jugar, excepto que la cantidad de equipos sea impar. En ese caso, todos los equipos, salvo ese, deben jugar en la última ronda. Esto implica un schedule más justo a nivel competitivo y deportivo.

$$\sum_{i \in E} \sum_{j \in E} x_{ijn} \geq \frac{|E| - 1}{2}. \tag{11}$$

12. Dominio de las variables:

$$\begin{aligned} z_{tk} &\in \{0,1\} \qquad \forall\, t \in T, \forall\, k \in F, \\ x_{ijk} &\in \{0,1\} \qquad \forall\, i,j \in E, i \neq j, \forall\, k \in F. \end{aligned}$$

La estructura del problema anteriormente descripto sería dificil de aplicar para resolver casos de *double round-robin* con más de 13 equipos.

5.2.3 Segunda Etapa del Modelo: asignación de días a los partidos

Una vez que se encuentra una solución satisfactoria que define el orden de los partidos del modelo de la primera etapa, los partidos se asignan a días específicos.

Input:

- el conjunto E definido en la etapa anterior.
- el conjunto $D = \{1, \ldots, m\}$ de días designados para que los partidos se jueguen, enumerados consecutivamente empezando por un Día 1 predefinido.
- el número g_i total de partidos que juega el equipo i.
- los conjuntos $J_i = \{1, \ldots, g_i\}$ de índices para el orden de los partidos que cada equipo i jugará determinado por el modelo de la primera etapa. Luego, para cada equipo, el índice p representa su partido número p.

El modelo contiene una variable x_{ipt} para cada equipo $i \in E$, cada partido $p \in J_i$ y cada fecha $t \in D$ tal que $x_{ipt} = 1$ si y sólo si el equipo i juega su p-ésimo partido en la fecha t. Además, contiene las variables auxiliares da_{ip} y db_{ip}, las cuales son necesarias para los casos en que no se pueda satisfacer las restricciones reflejan los requerimientos de la Asociación de Clubes que pide que haya dos días de descanso antes y después de cada *road trip*. Luego, $db_{ip} = 1$ si y sólo si el equipo

i tiene un solo día de descanso antes del primer partido de visitante de un *road trip* (el p-ésimo partido). Análogamente, $da_{ip} = 1$ si y sólo si el equipo i tiene un solo día de descanso después del último partido de visitante de un *road trip* (el p-ésimo partido). En la función objetivo, se aplica una penalidad cada vez que alguna de estas variables es 1.

Para expresar los días preferidos por los equipos, se define un parámetro $\text{pref}(i,t)$ para cada equipo $i \in E$ y cada fecha $t \in D$. Es igual a 1 si el equipo i prefiere jugar de local en la fecha t y 0 si no. Finalmente, se define un parámetro $\text{home}(i,p)$ para cada equipo $i \in E$ y cada partido $p \in J_i$ tal que $\text{home}(i,p) = 1$ si y sólo si el equipo i juega el partido p de local y 0 si no.

Luego, podemos definir formalmente el modelo de esta segunda etapa:

La función objetivo busca maximizar el número de partidos jugados en los días en que un equipo prefiere jugar de local y penaliza la cantidad de veces que los equipos empiezan o terminan un *road trip* sin el número requerido de días de descanso. Para alcanzar el efecto deseado y, al mismo tiempo, evitar que los tiempos de cómputo sean excesivos al calcular la solución, se aplica un valor de 1000 como coeficiente de penalidad:

$$\max \sum_{i \in E, t \in D} \sum_{p \in J_i} \text{pref}(i,t)\text{home}(i,p)x_{ipt} - 1000 \sum_{i \in E, p \in J_i} (db_{pt} + da_{ip}).$$

Las restricciones son las siguientes:

1. Se signa un día a cada partido.

$$\sum_{t \in D} x_{ipt} = 1 \qquad \forall\ i \in E, \forall\ p \in J_i. \tag{12}$$

2. El orden de los partidos no se invierte y no hay más de r días entre dos partidos consecutivos para cada equipo. Esta condición asegura que ningún equipo juegue dos partidos en el mismo día o en días consecutivos.

$$x_{ipt} \leq \sum_{s=t+2}^{\min(t+r,m)} x_{i,p+1,s} \qquad \forall i \in E, \forall p \in J_i, p < g_i, \forall t \in D, t \leq m-2. \tag{13}$$

El valor de r se fija en 7, para asegurar que no transcurra más de una semana entre dos partidos de cada equipo.

3. Ningún equipo juega de local el día que su estadio no está disponible.

$$x_{ipt} = 0 \qquad \forall\ i \in E, \forall\ p \in J_i, \forall\ t \in D, \tag{14}$$

Si el equipo i juega de local su partido p y su estadio no está disponible en la fecha t.

4. Dos equipos cuyos estadios están en la misma ciudad, no pueden jugar de local el mismo día.

$$x_{ipt} + x_{jqt} \leq 1 \qquad \forall\ i, j \in E, \forall\ p \in J_i, q \in J_j, \forall\ t \in D, \tag{15}$$

Si dos equipos diferentes i y j tienen sus estadios en la misma ciudad y juegan su partido p y q, respectivamente, de local.

5. Debe programarse un día de descanso entre dos partidos de visitante consecutivos en el mismo *road trip*.

$$x_{ipt} = x_{i,p+1,t+2} \qquad \begin{array}{l} \forall\ i \in E, \forall\ p \in J_i, p < g_i \\ \forall\ t \in D, t < m-1, \end{array} \tag{16}$$

donde p y $p+1$ son dos partidos consecutivos jugados por el equipo i en el mísmo *road trip*.

6. Queremos dejar al menos dos días de descanso antes y despues de cada *road trip*:

   (a) si p es el primer partido de un *road trip*,

$$x_{i,p-1,t-2} + x_{ipt} \leq 1 + db_{ip} \qquad \begin{array}{r} \forall\ i \in E, \forall\ p \in J_i, p > 1 \\ \forall\ t \in D, t > 2. \end{array} \tag{17}$$

(b) si p es el último partido de un *road trip*,

$$x_{ipt} + x_{i,p+1,t+2} \leq 1 + da_{ip} \qquad \forall\, i \in E, \forall\, p \in J_i, p < g_i \quad (18)$$
$$\forall\, t \in D, t < m - 1.$$

7. Cada partido entre dos equipos se asigna al mismo día. Esto debe ser incluido para asegurar la consistencia entre las fechas asignadas al mismo partido, dado que el modelo determina la fecha para cada partido por separado.

$$x_{ipt} = x_{jqt} \qquad \forall\, i, j \in E, \forall\, t \in D \quad (19)$$

si el partido p para el equipo i y el partido q para el equipo j es un partido en el cual i y j se enfrentan.

8. La televisación de los equipos: un conjunto $R \subset D$ de días se define como los días en los cuales un partido es televisado. La siguiente restricción incorpora los requerimientos de la televisadora que pide que al menos un partido se programe en cada día del conjunto:

$$\sum_{i \in E} \sum_{p \in J_i} x_{ipt} \geq 1 \qquad \forall t \in R.$$

9. Todos los equipos deben jugar el último día, excepto que la cantidad de equipos sea impar. En ese caso, todos los equipos menos uno deben jugar el último día.

$$\sum_{i \in E} x_{i,g_i,m} \geq \frac{|E| - 1}{2}. \quad (20)$$

10. Dominio de las variables:

$$\begin{aligned} x_{ipt} &\in \{0,1\} && \forall\, i \in E, \forall\, p \in J_i, \forall\, t \in D, \\ da_{ip} &\in \{0,1\} && \forall\, i \in E, \forall\, p \in J_i, \\ db_{ip} &\in \{0,1\} && \forall\, i \in E, \forall\, p \in J_i. \end{aligned}$$

A diferencia del modelo de la primera etapa, la estructura de este problema no le impide al solver encontrar una solución óptima para alguna instancia más grande. Lo que es más, encuentra una solución en algunos segundos para la mayoría de las instancias, y no tarda más de 11 minutos para las mas complejas. Sólo en pocas ocasiones las variables da_{it} y db_{it} tomaron valor 1.

5.2.4 Partición de Temporada

Dada la estructura de este problema, se vuelve difícil resolver casos de *double round-robin* con más de 13 equipos. Para poder sortear esta limitación para instancias con n mayor a 10, se divide la temporada en *bloques*.

En dichos casos, se utiliza las semanas de descanso a través de las *cutoff dates* para dividir los schedules en bloques separados pero no independientes. Esto implica que, una vez que se programa un partido en un determinado bloque, ese partido no puede ser programado en otro bloque diferente.

El modelo que resuelve cada bloque intenta maximizar el número de partiods jugados en cada bloque, para así, facilitar la resolución del resto de los bloques.

Cabe aclarar que los distintos bloques que conforman un torneo no necesariamente son resueltos secuencialmente. Con lo cual, se puede definir un orden no cronológico al permutar dichos bloques. La permutación para el caso cronológico, es decir que sigue el orden de los días del calendario, es σ como la función identidad.

Si la solución que se obtuvo mediante el algoritmo para el conjunto completo de bloques deja afuera algún partido, el procedimiento podría iterarse repitiendo los pasos aplicados a los bloques con restricciones adicionales, para generar soluciones diferentes para cada bloque. Por ejemplo, la asignación de un partido a una ronda determinada en la solución original para un bloque puede ser inhabilitada mediante una restricción, forzando una asignación diferente. Con lo cual, estamos ante un algoritmo de *backtracking* que correría hasta encontrar una solución que contenga todos los partidos.

5.2.5 Conclusiones

Como resultado se obtuvo una reducción de más del 30% del promedio de la distancia viajada por partido de visitante, lo cual trajo como beneficio menores costos para los equipos y la asociación, junto con menor fatiga por parte de los jugadores. Como consecuencia, los equipos pueden aplicar sus recursos a otras necesidades del club y del equipo, además de que los jugadores, al estar menos cansados, tienen un mejor desempeño deportivo individual y colectivamente. Esto trajo una reducción de US $ 700,000 por temporada en los costos globales de los equipos de ambas divisiones, además de que los jugadores atravesaron menos cansancio y fatiga durante el torneo.

Para algunos equipos, los montos de dinero ahorrados estuvieron por encima del promedio dada su ubicación lejos de los centros más importantes del país y de la buena elección de *road trips* solicitados a la liga como preferencia por parte los dirigentes de dichos clubes.

Varios equipos han usado ese dinero para financiarse los costos de pasajes aereos para *road trips* largos. Bajo los formatos anteriores de generación de *schedules*, muchos de estos viajes se realizaban en autobús. Este cambio significó una mejora sustancial en la experiencia de los jugadores a la hora de realizar viajes prolongados.

5.3 Aplicación a un Caso Real: Liga Nacional de Básquet de Chile

5.3.1 Introducción

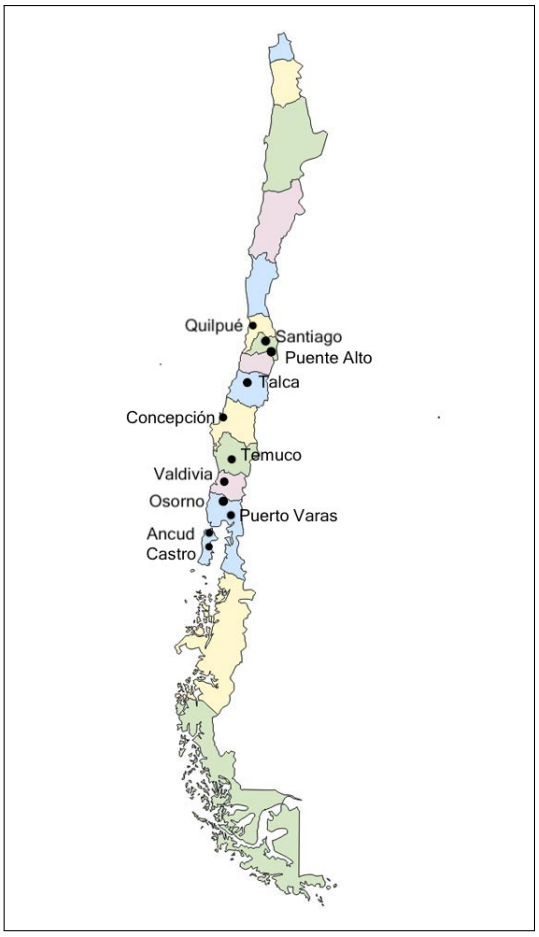
La Liga Nacional de Básquetbol (LNB), denominada por motivos de patrocinio como Liga DirecTV, es la principal competencia profesional de básquet de Chile, y es organizada por la Federación de Básquetbol de Chile.

El primer campeonato fue ganado por Español de Talca, club que además posee el récord de campeonatos con tres títulos. El actual campeón 2018-19 es Club Deportivo Valdivia, quien se impuso 4 a 1 en la serie final a Colegio Los Leones de Quilpué.

5.3.2 El Torneo

La temporada constará de dos fases: fase nacional y interzonal. Los 12 equipos que disputarán el campeonato jugarán 32 fechas. Cualquier equipo podrá descender a la segunda categoría del básquetbol nacional. En la fase nacional, cada equipo jugará todos contra todos en partidos de ida y vuelta. En esta misma etapa, se jugará en paralelo la fase zonal, llamado interzona, en los que jugarán todos contra todos en partidos de ida y vuelta. En primera rueda: Se medirán en encuentros de ida entre equipos de todo el país, esta fase consta de 11 fechas. En la fase interzona, los equipos jugarán partidos de ida y vuelta en su misma conferencia, generando un total de 10 fechas. En la segunda rueda, que también consta de 11 fechas, todos los equipos jugarán los partidos de vuelta. Todos los equipos pasarán a play offs.

5.3.3 Los Equipos



Conferencia Centro:

CD Colegio Los Leones (Quilpué)
CD Universidad Católica (Santiago)
CD Español de Talca
CD Universidad de Concepción
CD Municipal Puente Alto
CD AB Temuco

Conferencia Sur:

CD Las Ánimas de Valdivia
CD Osorno Básquetbol
CD Castro
CD AB Ancud
CD Valdivia
CD Atlético Puerto Varas

5.3.4 Desarrollo

Las ciudades donde se localizan los estadios de estos equipos son: Quilpué, Santiago de Chile, Puente Alto, Talca, Concepción, Temuco, Valdivia, Osorno, Puerto Varas, Ancud y Castro (ver mapa). Cabe mencionar que la mayor distancia entre estas ciudades es de 1068.54km, con lo cual, es propenso generar un *schedule* que intente minimizar la distancia total recorrida por los equipos al jugar de visitante.

La idea fue armar un fixture usando el Traveling Tournament Problem, basado en el modelo de programación lineal entera que se utilizó para el básquet de la liga nacional argentina [6], cuyo objetivo fue reducir la distancia total viajada por los equipos comparado con las temporadas anteriores usando también una selección de *road trips* predeterminadas elegidas por los equipos, tratando de hacer coincidir, lo más posible, el *schedule* final con los *road trips* deseados por los equipos y los denominados "razonables", geográficamente hablando. Además, se tuvo en cuenta restricciones más elementales:

- el lapso de días entre las fiestas (Navidad y Año Nuevo) no son días de posibles partidos.
- los días de la Copa Chile no se programan partidos de este torneo.

Como se mencionó anteriormente, el modelo está dividido en dos etapas sucesivas. La primera, define la secuencia en que cada equipo va a jugar contra todos los demás y, la segunda, asigna los días en los cuales se juega cada partido. Bajo estos parámetros, los partidos se juegan en cualquier día de la semana y los partidos consecutivos de visitante son programados en *road trips*.

Nos concentraremos en lograr la minimización de la distancia recorrida por los 12 equipos en toda la fase nacional. Se propusieron *road trips* para que los equipos realicen, basados en el fixture que se utilizó en el torneo de la temporada 2018/19 y en otras que se creyeron convenientes en base a la cercanía entre los equipos.

5.3.5 Resultados

A continuación se presentan el fixture oficial del torneo 2018/19 y el fixture resultante del Traveling Tournament Problem.

Fixture Oficial Liga DirecTV 2018/19

	Sa	Do	Lu	Ma	Mi	Ju	Vi	Sa	Do
	17-nov	18-nov	19-nov	20-nov	21-nov	22-nov	23-nov	24-nov	25-nov
Leones de Quilpué	**at Castro**	**at Ancud**						Osorno	Las Ánimas
Español de Talca	**at Las Animas**	**at Osorno**						Concepción	Temuco
AB Temuco	Valdivia							**at Puenet Alto**	**at Talca**
Universidad Católica	**at Ancud**	**at Castro**						Las Ánimas	Osorno
Universidad de Concepción	Puerto Varas	Valdivia						**at Español de Talca**	**at Puente Alto**
Municipal Puente Alto	**at Osorno**	**at Las Ánimas**						Temuco	Concepción
CD Valdivia	**at Temuco**	**at Concepción**						Deportes Castro	Ancud
AB Ancud	Católica	Leones de Quilpué						**at Puerto Varas**	**at Valdivia**
Osorno Básquetbol	Puente Alto	Español de Talca						**at Quilpué**	**at Católica**
Deportes Castro	Leones de Quilpué	Católica						**at Valdivia**	**at Puerto Varas**
Las Ánimas	Español de Talca	Puente Alto						**at Católica**	**at Quilpué**
Atlético Puerto Varas	**at Concepción**							Ancud	Castro

Vi	Sa	Do	Lu	Ma	Mi	Ju	Vi	Sa	Do	Lu
7-dic	8-dic	9-dic	10-dic	11-dic	12-dic	13-dic	14-dic	15-dic	16-dic	17-dic
	at Talca	**at Puente Alto**							**at Concepción**	**at Temuco**
	Leones de Quilpué	Católica						**at Valdivia**	**at Puerto Varas**	
	Ancud	Castro							Católica	Quilpué
at Puente Alto		**at Talca**						**at Concepción**	**at Temuco**	
	Castro	Ancud						Católica	Quilpué	
Católica		Leones de Quilpué						**at Puerto Varas**	**at Valdivia**	
	at Osorno	**at Las Ánimas**						Talca	Puente Alto	
	at Temuco	**at Concepción**						Osorno	Las Ánimas	
	Valdivia	Puerto Varas						**at Ancud**	**at Castro**	
	at Concepción	**at Temuco**						Las Ánimas	Osorno	
	Puerto Varas	Valdivia						**at Castro**	**at Ancud**	
	at Las Ánimas	**at Osorno**						Puente Alto	Talca	

Vi	Sa	Do	Lu	Ma	Mi	Ju	Vi	Sa	Do
21-dic	22-dic	23-dic	24-dic	25-dic	26-dic	27-dic	28-dic	29-dic	30-dic
Puerto Varas	Valdivia								
Ancud	Castro								
at Las Ánimas	**at Osorno**								
Valdivia	Puerto Varas								
at Osorno	**at Las Ánimas**								
Castro	Ancud								
at Católica	**at Quilpué**								
at Talca	**at Puente Alto**								
Concepción	Temuco								
at Puente Alto	**at Talca**								
Temuco	Concepción								
at Quilpué	**at Católica**								
			Fiestas						

Sa	Do	Lu	Ma	Mi	Ju	Vi	Sa	Do
5-ene	6-ene	7-ene	8-ene	9-ene	10-ene	11-ene	12-ene	13-ene
at Católica	Católica						Castro	Ancud
Puente Alto	**at Puente Alto**						Las Ánimas	Osorno
Concepción		**at Concepción**					**at Valdivia**	**at Puerto Varas**
Quilpué	**at Quilpué**						Ancud	Castro
at Temuco		Temuco					**at Puerto Varas**	**at Valdivia**
at Talca	Talca						Osorno	Las Ánimas
Puerto Varas	**at Puerto Varas**						Temuco	Concepción
at Castro	Castro						**at Católica**	**at Quilpué**
Las Ánimas	**at Las Ánimas**						**at Puente Alto**	**at Talca**
Ancud	**at Ancud**						**at Quilpué**	**at Católica**
at Osorno	Osorno						**at Talca**	**at Puente Alta**
at Valdivia	Valdivia						Concepción	Temuco

Sa	Do	Lu	Ma	Mi	Ju	Vi	Sa	Do
19-ene	20-ene	21-ene	22-ene	23-ene	24-ene	25-ene	26-ene	27-ene
at Osorno	**at Las Ánimas**						Talca	Puente Alto
at Concepción	**at Temuco**						**at Quilpué**	Católica
Puente Alto	Talca						**at Ancud**	**at Castro**
at Las Ánimas	**at Osorno**						Puente Alto	Talca
Talca	Puente Alto						**at Castro**	**at Ancud**
at Temuco	**at Concepción**						**at Católica**	**at Quilpué**
at Castro	**at Ancud**			Osorno	Las Ánimas			
Puerto Varas	Valdivia						Temuco	Concepción
Quilpué	Católica			**at Valdivia**				**at Puerto Varas**
Valdivia	Puerto Varas						Concepción	Temuco
Católica	Quilpué				**at Valdivia**		**at Puerto Varas**	
at Ancud	**at Castro**						Las Ánimas	Osorno

Mi	Ju	Vi	Sa	Do	Lu	Ma	Mi	Ju
30-ene	31-ene	1-feb	2-feb	3-feb	5-feb	6-feb	9-feb	10-feb
			Temuco	Concepción			**at Puerto Varas**	**at Valdivia**
			Valdivia	Puerto Varas			**at Ancud**	**at Castro**
			at Quilpué	**at Católica**			Las Ánimas	
			Concepción	Temuco			**at Valdivia**	**at Puerto Varas**
			at Católica	**at Quilpué**				Las Ánimas
			Puerto Varas	Valdivia			**at Castro**	**at Ancud**
			at Talca	**at Puente Alto**			Católica	Quilpué
					at Osorno	**at Las Ánimas**	Talca	Puente Alto
					Ancud	Castro		
					at Las Ánimas	**at Osorno**	Puente Alto	Talca
					Castro	Ancud	**at Temuco**	**at Concepción**
			at Puente Alto	**at Talca**			Quilpué	Católica
	ventanaFiba5							

Fixture Resultante del Traveling Tournament Problem

	Ju	Vi	Sa	Do	Lu
	1–Sept	2–Sept	3–Sept	4–Sept	5–Sept
Leones de Quilpué					
Español de Talca					
AB Temuco					
Universidad Católica		@CD Valdivia		@Atlético Puerto Varas	
Universidad de Concepción					@Osorno Básquetbol
Municipal Puente Alto					
CD Valdivia		Universidad Católica			
AB Ancud					
Osorno Básquetbol			Deportes Castro		Universidad de Concepción
Deportes Castro			@Osorno Básquetbol		
Las Ánimas					
Atlético Puerto Varas				Universidad Católica	

Ma	Mi	Ju	Vi	Sa	Do	Lu	Ma
6–Sept	7–Sept	8–Sept	9–Sept	10–Sept	11–Sept	12–Sept	13–Sept
				Español de Talca		CD Valdivia	
Las Ánimas				@Leones de Quilpué		Atlético Puerto Varas	
					@AB Ancud		@Deportes Castro
				CD Valdivia			
		Las Ánimas		Atlético Puerto Varas			
AB Ancud				@Universidad Católica		@Leones de Quilpué	
@CD Valdivia					AB Temuco		
							AB Temuco
@Español de Talca		@Municipal Puente Alto					
				@Municipal Puente Alto		@Español de Talca	

Mi	Ju	Vi	Sa	Do	Lu	Ma
14–Sept	15–Sept	16–Sept	17–Sept	18–Sept	19–Sept	20–Sept
	AB Temuco					@Las Ánimas
	@AB Ancud		@Deportes Castro			
	@Leones de Quilpué		@Universidad Católica			
			AB Temuco			
			CD Valdivia			
	@Deportes Castro		@AB Ancud			
			@Universidad de Concepción			Deportes Castro
	Español de Talca		Municipal Puente Alto			
	Las Ánimas		@Las Ánimas			Atlético Puerto Varas
	Municipal Puente Alto		Español de Talca			@CD Valdivia
	@Osorno Básquetbol		Osorno Básquetbol			Leones de Quilpué
						@Osorno Básquetbol

Mi	Ju	Vi	Sa	Do	Lu
21–Sept	22–Sept	23–Sept	24–Sept	25–Sept	26–Sept
		AB Ancud			
		Universidad Católica		@Municipal Puente Alto	
Universidad de Concepción					Deportes Castro
AB Ancud		@Español de Talca			
@AB Temuco			Deportes Castro		Osorno Básquetbol
				Español de Talca	
					Las Ánimas
@Universidad Católica		@Leones de Quilpué			
					@Universidad de Concepción
	@Atlético Puerto Varas		@Universidad de Concepción		@AB Temuco
					@CD Valdivia
	Deportes Castro				

Mi	Ju	Vi	Sa	Do	Lu	Ma
28–Sept	29–Sept	30–Sept	1–Oct	2–Oct	3–Oct	4–Oct
	@Universidad Católica		Atlético Puerto Varas		Las Ánimas	
Municipal Puente Alto						
	@Universidad de Concepción					
	Leones de Quilpué		Las Ánimas		Atlético Puerto Varas	
	AB Temuco			@Municipal Puente Alto		
@Español de Talca				Universidad de Concepción		
				@Deportes Castro		@AB Ancud
	Atlético Puerto Varas			Osorno Básquetbol		CD Valdivia
				@AB Ancud		@Deportes Castro
				CD Valdivia		Osorno Básquetbol
			@Universidad Católica		@Leones de Quilpué	
	@AB Ancud		@Leones de Quilpué		@Universidad Católica	

Mi	Ju	Vi	Sa	Do	Lu	Ma
5–Oct	6–Oct	7–Oct	8–Oct	9–Oct	10–Oct	11–Oct
					@Español de Talca	
AB Temuco					Leones de Quilpué	
@Español de Talca		@Municipal Puente Alto				Universidad Católica
				@Universidad de Concepción		@AB Temuco
				Universidad Católica		
		AB Temuco				
		@Osorno Básquetbol				
					@Atlético Puerto Varas	
		CD Valdivia				
					@Las Ánimas	
Atlético Puerto Varas					Deportes Castro	
@Las Ánimas					AB Ancud	

AQ	AR	AS	AT	AU	AV
Mi	Ju	Vi	Sa	Do	Lu
12–Oct	13–Oct	14–Oct	15–Oct	16–Oct	17–Oct
@Municipal Puente Alto		@Universidad de Concepción		@AB Temuco	
@Atlético Puerto Varas					
		AB Ancud		Leones de Quilpué	
				Municipal Puente Alto	
		Leones de Quilpué		AB Ancud	
Leones de Quilpué				@Universidad Católica	
		@Atlético Puerto Varas		Osorno Básquetbol	
@Las Ánimas		@AB Temuco		@Universidad de Concepción	
				@CD Valdivia	
AB Ancud				@Atlético Puerto Varas	
Español de Talca		CD Valdivia		Las Ánimas	

Ma	Mi	Ju	Vi	Sa	Do	Lu	Ma
18–Oct	19–Oct	20–Oct	21–Oct	22–Oct	23–Oct	24–Oct	25–Oct
@Las Ánimas		Osorno Básquetbol					
			CD Valdivia				
Osorno Básquetbol							
			@AB Temuco				
@Municipal Puente Alto		@Español de Talca					
Español de Talca							
					copa Chile	copa Chile	

Mi	Ju	Vi	Sa	Do	Lu
26–Oct	27–Oct	28–Oct	29–Oct	30–Oct	31–Oct
	@CD Valdivia		@Osorno Básquetbol		
@Municipal Puente Alto				Universidad de Concepción	
@Deportes Castro		@AB Ancud		@Universidad Católica	
Universidad Católica					Deportes Castro
	AB Temuco				
		Universidad de Concepción			
			AB Temuco		
Universidad de Concepción		Atlético Puerto Varas			@Municipal Puente Alto
		@Deportes Castro			

Ma	Mi	Ju	Vi	Sa	Do
1–Nov	2–Nov	3–Nov	4–Nov	5–Nov	6–Nov
Universidad de Concepción					Osorno Básquetbol
	Deportes Castro				AB Ancud
@Leones de Quilpué					
		@Atlético Puerto Varas		@CD Valdivia	
	@Las Ánimas			Municipal Puente Alto	
					@Español de Talca
					@Leones de Quilpué
	@Español de Talca				
	CD Valdivia				
		Municipal Puente Alto			

Ma	Mi	Ju	Vi	Sa	Do	Lu	Ma
8–Nov	9–Nov	10–Nov	11–Nov	12–Nov	13–Nov	14–Nov	15–Nov
		@Atlético Puerto Varas					
		@Universidad Católica					
Las Ánimas							
Osorno Básquetbol		Español de Talca					
		Las Ánimas					
AB Ancud							
@Municipal Puente Alto		@Deportes Castro					
@Universidad Católica							
		AB Ancud					
@AB Temuco		@Universidad de Concepción					
		Leones de Quilpué					
				ventanaFiba5			ventanaFiba5

Ju	Vi	Sa	Do	Lu	Ma	Mi
17–Nov	18–Nov	19–Nov	20–Nov	21–Nov	22–Nov	23–Nov
@Osorno Básquetbol		@CD Valdivia				
				Universidad de Concepción		
				Atlético Puerto Varas		@Las Ánimas
		@Las Ánimas		@Osorno Básquetbol		
		Atlético Puerto Varas		@Español de Talca		
		Leones de Quilpué				
					Deportes Castro	
Leones de Quilpué				Universidad Católica		
					@AB Ancud	
		Universidad Católica				AB Temuco
		@Universidad de Concepción		@AB Temuco		

Ju	Vi	Sa	Do	Lu	Ma	Mi
24–Nov	25–Nov	26–Nov	27–Nov	28–Nov	29–Nov	30–Nov
Deportes Castro						Municipal Puente Alto
CD Valdivia						
		Osorno Básquetbol				
		Deportes Castro				
	@Las Ánimas					
		CD Valdivia				@Leones de Quilpué
@Español de Talca		@Municipal Puente Alto				
						Las Ánimas
		@AB Temuco				
@Leones de Quilpué		@Universidad Católica				
	Universidad de Concepción					@AB Ancud

Ju	Vi	Sa	Do	Lu
1–Dic	2–Dic	3–Dic	4–Dic	5–Dic
			Universidad Católica	
	@Universidad de Concepción		@AB Temuco	
	Municipal Puente Alto		Español de Talca	
			@Leones de Quilpué	
	Español de Talca		Municipal Puente Alto	
	@AB Temuco		@Universidad de Concepción	
			Atlético Puerto Varas	
			@Osorno Básquetbol	
@Atlético Puerto Varas			AB Ancud	
	Las Ánimas			
	@Deportes Castro			
Osorno Básquetbol			@CD Valdivia	

Ma	Mi	Ju	Vi
6–Dic	7–Dic	8–Dic	9–Dic
@Deportes Castro		@AB Ancud	
@CD Valdivia		@Osorno Básquetbol	
			@Atlético Puerto Varas
@AB Ancud		@Deportes Castro	
	@Atlético Puerto Varas		@CD Valdivia
@Osorno Básquetbol		@Las Ánimas	
Español de Talca			Universidad de Concepción
Universidad Católica		Leones de Quilpué	
Municipal Puente Alto		Español de Talca	
Leones de Quilpué		Universidad Católica	
		Municipal Puente Alto	
	Universidad de Concepción		AB Temuco

El fácil observar que el fixture oficial es el tradicional por parejas. Se establecen ciertas parejas de equipos, y cada una visita a otra, dos días seguidos que suelen ser sábado y domingo. Éste método suele ser eficiente y, dada la disposición geográfica de Chile y la localización de los equipos participantes del torneo, el fixture se puede construir a mano. El fixture oficial resulta en una distancia total recorrida por todos los equipos de 68795km durante todo el torneo, mientras que nuestro fixture resultante implica una distancia total de 67485km. Esto representa una mejora del 2% a una solución buena. El modelo fue corrido en una computadora con un procesador 2,5 GHz Intel Core i7 y se utilizó CPLEX 12.8 para resolver el modelo de programación entera.

Cabe aclarar que en nuestro caso no alcanzamos el óptimo por lo que hay margen, en un trabajo futuro, para conseguir mejorar soluciones; por ejemplo, fortaleciendo las formulaciones de los modelo de programación entera, incorporando nuevos conjuntos de giras o directamente consiguiendo computadoras más veloces.

Otra posibilidad de trabajo de investigación futuro en relación a este problema sería el desarrollo de una herurística que mejore en mayor porcentaje la solución obtenida con nuestro algoritmo. Heurísticas del tipo *tabu search* han demostrado ser eficientes para resolver este tipo de problemas, como por ejemplo, se hizo en el paper *An Application of the Traveling Tournament Problem: The Argentine Volleyball League* [3]. En ese trabajo se describe un proceso de optimización usado para crear el schedule de la Primera División de la Liga Profesional de Voleibol argentina. Los equipos de la liga fueron agrupados por parejas, y los partidos se disputan jueves y sábado. En cada par de partidos de jueves y sábado consecutivos, los dos equipos de cada pareja juegan contra dos equipos de otra pareja. En este caso también se busca minimizar la distancia que los equipos deben viajar, ya que las ciudades de cada uno de ellos se encontraban relativamente alejadas unas de otras a lo largo de la gran superficie geográfica de Argentina y los equipos no vuelven a sus ciudades de origen entre partidos consecutivos de visitante. Este formato de parejas despierta dos cuestiones cruciales:

- cómo agrupar en parejas a los equipos
- cómo programar los partidos

Con lo cual, se aplicaron técnicas de programación entera y búsqueda tabú para resolver dichas cuestiones.

6 Conclusiones

Podríamos decir que el Traveling Tournament Problem es una clase de problemas no sólo interesante para la investigación operativa, sino también, para la comunidad deportiva que se beneficia económica y deportivamente hablando.

Es muy amplio el abanico de posibilidades para el TTP. Las diversas técnicas, tipos y heurísticas descriptas en ésta tesis son sólo algunas de las aplicaciones y approaches de este tipo de problema de sports scheduling. Con lo cual, no sólo se cubren muchas áreas de la investigación operativa y la optimización: teoría de grafos, programación lineal entera, algoritmos genéticos, etc; sino que también se abarcan varios tipos de deportes practicados en diferentes partes del mundo. Esto implica múltiples puertas para abrir, donde uno siempre puede encontrar una aplicación de alguna forma distinta a la anterior y, también, igual o más desafiante.

Personalmente, siempre me apasionó el deporte, lo cual hace al TTP aún más interesante para mí y, desde que me topé en la carrera con esta área, supe que era una de mis favoritas. Es claro que significa una veta de divulgación importantísima, una forma de hacer ciencia mientras se atrae gente ajena a la comunidad científica: asociaciones y clubes deportivos, universidades, escuelas, etc. Es una forma de contarle a la gente que la matemática está presente en muchas de las cosas de la vida cotidiana, y así, acercarla a la ciencia. Es por eso que veo muy importante y desafiante la investigación por parte de la Facultad de Ciencias Exactas y Naturales sobre estos temas, por ejemplo, el trabajo de sports scheduling que hizo el grupo de investigación del Instituto del Cálculo para las ligas de fútbol, básquet y voley de la Argentina. Haciendo de la matemática una ciencia atractiva para los medios y, así, para la sociedad.

La aplicación del Traveling Tournament Problem para generar un fixture óptimo para diferentes y múltiples ligas deportivas de todo el mundo, trae un ahorro económico en cuanto a gastos para traslados de los equipos, un aumento en las ventas de entradas a los partidos en los estadios, disminución de fatiga por parte de los jugadores al tener menos horas de viaje durante el torneo, lo cual les ayuda a tener una mejor performance en los partidos, además de que genera una mayor equidad para todos los equipos, grandes o chicos, ya que todos ellos viajarían distancias parecidas durante todo el torneo, sin que se generen ventajas para algunos.

Esta interesante y peculiar clase de problemas tiene la mayoría de sus instancias abiertas a resolución, ya que presenta un gran desafío en cuanto a su complejidad computacional. Varios de los papers citados aquí muestran la importancia de com-

binar técnicas diferentes para atacar este problema ya que combina desafíos en cuanto a la factibilidad y la optimalidad.

Uno de los objetivos de Easton, Nemhauser y Trick al proponer al Traveling Tournament Problem como problema de referencia en la investigación operativa, es fomentar la investigación sobre nuevas técnicas para resolver este problema y las variaciones del mismo.

Una de las mayores dificultades de este problema es aumentar las restricciones a medida que aumenta la cantidad de equipos mientras se intenta controlar, o no prolongar mucho más, el tiempo de corrida computacional del algoritmo. Una de las restricciones que se podrían intentar satisfacer son, por ejemplo, limitar el número de fechas libres (byes) que cada equipo puede tener como máximo en el Relaxed Traveling Tournament Problem.

Otras futuras investigaciones sobre el Traveling Tournament Problem podrían dedicarse a estudiar su complejidad. Se ha mostrado en este trabajo que sin la restricción sobre la cantidad de partidos consecutivos de local y visitante permitidos, este problema es NP-hard. Con su trabajo, Bhattacharyya espera que su método sea útil para probar que el Traveling Tournament Problem es hard a partir de situaciones prácticas donde, por ejemplo, U se fija en 3.

7 Referencias

[1] JC Bean, JR Birge, Reducing travelling costs and player fatigue in the national básquet association, *Interfaces*, Vol. 10, No. 3, páginas 98-102, 1980

[2] R Bhattacharyya, Complexity of the unconstrained traveling tournament problem *Operations Research Letters*, 44 (5), páginas 649–654, 2016.

[3] F Bonomo, A Cardemil, G Durán, J Marenco, D Sabán, An Application of the Traveling Tournament Problem: The Argentine Volleyball League, *Interfaces* 42(3): páginas 245-259, 2012.

[4] D Briskorn, Sports Leagues Scheduling: Models, Combinatorial Properties, and Optimization Algorithms, *XII*, 164 páginas, 2007

[5] RT Campbell, DS Chen, A minimum distance basketball scheduling problem, *Management science in sports*, páginas 15–26, 1976.

[6] D Costa, An evolutionary tabu search algorithm and the NHL scheduling problem, *Information Systems and Operational Research*, Vol. 33, páginas 161-178, 1995

[7] G Durán, S Durán, J Marenco, F Mascialino, P Rey, Scheduling Argentina's professional básquet leagues: A variation on the relaxed Travelling Tournament Problem, *European Journal of Operational Research* 275(3), páginas 1126-1138, 2018

[8] K Easton, G Nemhauser, M Trick, The traveling tournament problem: Description and benchmarks, In *Seventh International Conference on the Principles and Practice of Constraint Programming (CP 99)*, volume 2239 of LNCS, páginas 580-589, Springer-Verlag, 2001

[9] K Easton, G Nemhauser, M Trick, Solving the travelling tournament problem: A combined integer programming and constraint programming approach,

in *Proc. of the 4th Int. Conf. on the Practice and Theory of Automated Timetabling* (PATAT-2002), volumen 2740 de LNCS, páginas 100-109, 2003

[10] JA Ferland, C Fleurent, Computer aided scheduling for a sport league, *INFOR 29*, páginas 14-25, 1991.

[11] W Fraser, The role of computer simulation in building the National Hockey League schedule, IBM Canada Limited Montreal Quebec Canada, 1982.

[12] D Froncek, Scheduling the Czech national basketball league, Congressus Numerantium, 153, páginas 5-24, 2001.

[13] M Henz, Scheduling a major college Basketball conference, *Operations Research* volume 19(1), páginas 163-168, 2001.

[14] R Hoshino, K Kawarabayashi, A multi-round generalization of the traveling tournament problem and its application to Japanese baseball, *European Journal of Operational Research* (submitted), 2010.

[15] P Rasmussen, M Trick, A Benders approach for the constrained minimum break problem, *European Journal of Operational Research* 177:198–213, 2007.

[16] R Bao, Time Relaxed Round Robin Tournament and the NBA Scheduling Problem, Master's thesis, Cleveland State University, 2006.

[17] Schonberger, Memetic Algorithm timetabling for non-commercial sport leagues, *European Journal of Operational Research* 153, 2004

[18] C Thielen, S Westphal, Approximation algorithms for TTP(2), *Mathematical Methods of Operations Research*, 76(1):1–20, 2012.

[19] M Trick, G Nemhauser, scheduling a major college Básketball conference, *Operations Research* volume 19(1), páginas 163-168, 2001

[20] T Voorhis, Highly constrained college básquetball scheduling, *Journal of the Operational Research Society* 53, páginas 603-309, 2002.

[21] D Werra, Geography, Games and Graphs, *Discrete Applied Mathematics*, 2, páginas 327-337, 1980.

[22] D Werra, Minimizing irregularities in sports schedules using graph theory. *Discrete Applied Mathematics*, 4, páginas 217-226, 1982.

[23] S Westphal, Scheduling the German Basketball League, Interfaces, 44, páginas 498–508, 2014

[24] R Willis, B Terrill, Scheduling the Australian state cricket season using simulated annealing, *Journal of the Operational Research Society*, 45 (3), páginas 276-280, 1994

[25] M Wright, Timetabling County Cricket Fixtures Using a Form of Tabu Search, *Journal of the Operational Research Society*, 45 (7), páginas 758–770, 1995

[26] M Wright, Scheduling fixtures for basketball New Zealand, Computers Operations Research, 3, páginas 1875–1893, 2006

[27] M Xiao, S Kou, An Improved Approximation Algorithm for the Traveling Tournament Problem with Maximum Trip Length Two, Conference Paper, 41st International Symposium on Mathematical Foundations of Computer Science 2016.