Análisis y Diseño de Algoritmos II

El Problema de la Planificación del Calendario de Torneos Deportivos

Edinson Orlando Dorado Dorado
edinson.dorado@correounivalle.edu.co
1941966

Alessandro Díaz García
alessandro.diaz@correounivalle.edu.co
1940983

Junio de 2023

1. El modelo

El proyecto consiste en la planificación del calendario de un torneo deportivo, para nuestro caso, el problema cuenta con las siguientes características básicas:

- El número de equipos (n) es par
- Un partido enfrenta dos equipos, uno de ellos es local y el otro visitante.
- Una fecha es un conjunto de partidos donde cada equipo juega una y solo una vez
- Un torneo todos contra todos ida y vuelta ($tttiv$) es un torneo en el que cada par de equipos juega dos partidos entre ellos: el de ida y el de vuelta.
- Un calendario a dos rondas es un conjunto de fechas tal que en el conjunto de todos los partidos cada equipo se enfrenta exactamente dos veces a cada otro equipo; en estos dos enfrentamientos los equipos se alternan la localía. Al primer partido entre dos equipos se le llama el partido de ida y al segundo se le llama el partido de vuelta.
- Un calendario a dos rondas para un $tttiv$ consiste de $2(n-1)$ fechas, cada una conteniendo $n/2$ partidos.

Para representar el calendario se usará una matriz Cal de dimensión:

$$2(n-1) \times n \tag{1}$$

tal que $Cal[i,j] = k$ significa que en la fecha i el equipo j juega de local contra el equipo k y $Cal[i,j] = -k$ significa que en la fecha i el equipo j juega de visitante contra el equipo k.

Las distancias entre las ciudades sedes de cada equipo se representan por medio de una matriz D de dimensión $n*n$: Cuando el equipo i juega de visitante ante el equipo j debe desplazarse una distancia d_{ij} : Si en el calendario el equipo i juega k partidos consecutivos de visitante ante los equipos j_1,..., j_k, se dice que el equipo i está de gira. La gira es de tamaño k y su costo es $d_{ij_1} + d_{j_1j_2} + d_{j_2j_3} + ... + d_{j_{k-1}j_k} + d_{j_ki}$

Por otro lado, si en el calendario el equipo i juega k partidos consecutivos de local se dice que el equipo i está en permanencia de tamaño k.

1.1. Descripción del modelo

El modelo propuesto para resolver este problema es un modelo de programación lineal entera, esto debido a que las variables de decisión de la matriz del calendario Cal, se definen como variables enteras, y debido a que las restricciones y la función objetivo se expresan mediante ecuaciones y desigualdades lineales, es lineal.

```
int: n;  % Numero de equipos
int: Min;  % Tamano minimo de las giras y permanencias
int: Max;  % Tamano maximo de las giras y permanencias
array[1..n, 1..n] of int: D; % Matriz de distancias

% Variables de decision
array[1..2*(n-1), 1..n] of var -n..n: Cal;  % Matriz Calendario

% Restricciones
% ...

% Calculo de la suma del costo de las distancias
totalDistance = sum(i in 1..2*(n-1), j in 1..n) (
    let {
              ...
    } in value
);

% Objetivo: Minimizar la distancia total de las giras
solve minimize totalDistance;
```

1.1.1. Parámetros de entrada

n = Número de equipos
Min = Tamaño mínimo de las giras y permanencias
Max = Tamaño máximo de las giras y permanencias
D = Matriz de distancias de tamaño $n*n$ que indica el costo de las distancias entre las ciudades de los equipos.

1.1.2. Variables de decisión

El modelo requiere de $2 * (n-1) * n$ variables enteras de decisión, que son representadas en la matriz de solución Cal.

$Cal[i,j]$=Equipo oponente con quien juego de local o visitante en una fecha i, siendo el equipo j

1.1.3. Restricciones del problema

Todas las restricciones son enteras lineales, debido a que todas las restricciones se formulan en términos de ecuaciones o desigualdades lineales y todas las variables están restringidas a tomar valores enteros.

- $Cal[i,j] = k$ si y solo si $Cal[i,k] = -j$
- En cada fecha i, todos los equipos deben jugar contra los demás equipos, sin repeticiones. Cada equipo debe tener un único oponente en cada fecha y se cumple el requisito de jugar contra todos los demás equipos.

$$\{|\text{Cal}[i,j]| : 1 \leq j \leq n\} = \{1, 2, \ldots, n\}, \quad \forall 1 \leq i \leq 2(n-1)$$

- En cada fecha i, la cantidad de partidos donde un equipo es local debe ser igual a la cantidad de partidos donde ese mismo equipo es visitante. Esto garantiza que haya un equilibrio en el número de partidos jugados como local y visitante para cada equipo

$$\frac{n}{2} = |\{Cal[i,j] > 0 : 1 \leq j \leq n\}| = |\{Cal[i,j] < 0 : 1 \leq j \leq n\}|,$$

  para todo $1 \leq i \leq 2(n-1)$.

- Para cada par de equipos diferentes j y k, existen dos fechas i_1 e i_2 en las que el equipo j juega como local contra el equipo k en una fecha y como visitante en la otra fecha.

$$\forall j \in [1..n], \forall k \neq j, \exists i_1, i_2 \in [1, 2(n-1)] : \text{Cal}[i_1, j] = k \text{ y } \text{Cal}[i_2, j] = -k$$

- Las giras y permanencias deben ser como mínimo Min fechas consecutivas y como máximo Max fechas consecutivas

$$\forall j \in \{1, 2, \ldots, n\}, i \in \{1, 2, \ldots, 2(n-1) - \text{Max}\} :$$

$$consecutivePos = \sum_{k=i}^{i+Max} \text{bool2int}(\text{Cal}[k,j] > 0)$$

$$consecutiveNeg = \sum_{k=i}^{i+Max} \text{bool2int}(\text{Cal}[k,j] < 0)$$

$$(consecutivePos \geq Min \wedge consecutivePos \leq Max) \wedge$$

$$(consecutiveNeg \geq Min \wedge consecutiveNeg \leq Max)$$

- Esta restricción establece que para todas las fechas, excepto la primera, y para cada equipo, los partidos consecutivos no pueden tener el mismo valor absoluto. Es decir, el equipo no puede tener partidos consecutivos con el mismo oponente.

$$\forall i \in \{2, 3, \ldots, 2(n-1)\}, j \in \{1, 2, \ldots, n\} : \quad |\text{Cal}[i-1, j]| \neq |\text{Cal}[i, j]|$$

1.1.4. Función objetivo

Minimizar el costo de la suma de todas las giras de los equipos:
$d_{ij_1} + d_{j_1j_2} + d_{j_2j_3} + ... + d_{j_{k-1}j_k} + d_{j_ki}$

$$Minimize: \quad \text{totalDistance} = \sum_{i=1}^{2(n-1)} \sum_{j=1}^{n} D(|\text{Cal}[i-1, j]|, |\text{Cal}[i, j]|),$$

Este modelo es adecuado para el problema planteado debido a que cumple con todas las restricciones exigidas por el problema y considera los valores de las distancias entre las distintas ciudades buscando minimizar el costo total de las giras en el calendario y así lograr encontrar soluciones óptimas o cercanas a óptimas.

2. Implementación

Las bondades que consideramos que presenta nuestra implementación están en que captura las restricciones básicas del problema impuestas en el enunciado y que busca minimizar la distancia total recorrida por los equipos, buscando llegar al óptimo o al menos satisfacer los requerimientos que tiene el problema.

Algunas de las falencias o mejoras que puede tener nuestra implementación, son que no incluimos la restricción opcional de que no se deben programar partidos de vuelta hasta tanto no se hayan programado todos los partidos de ida.

La eficiencia y utilidad en la práctica depende de la cantidad de equipos que tenga una instancia de un problema, ya que para un número de equipos igual o mayor a 20, el programa puede tardar más de 30 minutos en retornar una solución, de acuerdo a nuestras pruebas realizadas.

Para ello pensamos en una restricción adicional que creemos que no es muy probable que afecte las posibles soluciones de los problemas. Esta consiste en que en la solución se debe asegurar que por lo menos la mitad de los equipos tengan al menos una permanencia igual a Max. Habíamos realizado una prueba con el problema de la batería de pruebas de nombre **case20-3** y antes de esta restricción adicional estuvo más de 2 horas en ejecución y no nos retornó ninguna solución, pero con esta nueva restricción obtuvimos una solución en 1 hora, aunque no es poco tiempo, pensamos que mejoró bastante el desempeño.

La optimalidad del modelo es adecuada, ya que responde a las restricciones impuestas en el problema, toma en cuenta los parámetros de entrada adecuados

y busca minimizar la función objetivo de acuerdo a las variables de decisión. El problema está sujeto a limitaciones prácticas como el número de equipos del problema y los recursos computaciones que tengamos disponibles. Consideramos que puede ser óptimo desde un punto de vista teórico, pero en la práctica un número de equipos muy grande puede afectar la optimalidad ya que puede que no se garantice encontrar un óptimo en un tiempo razonable.

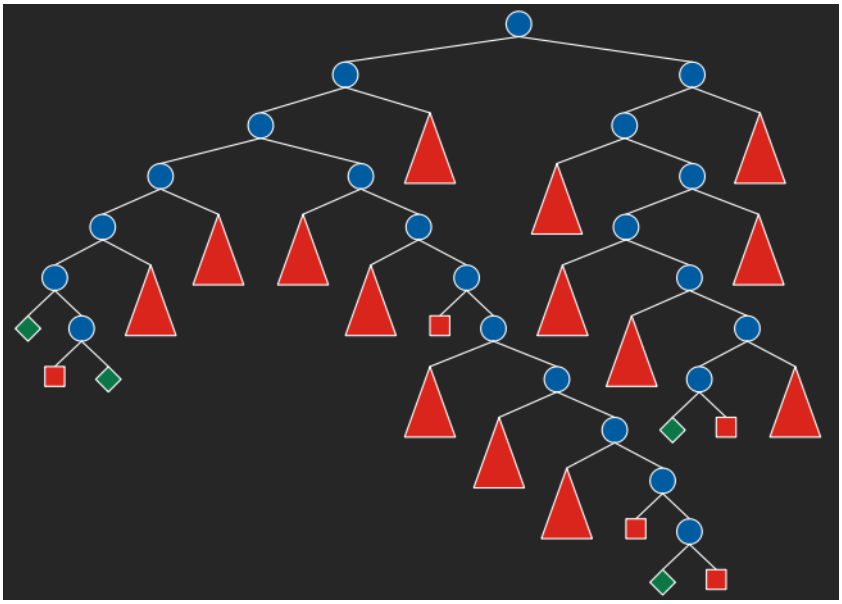
3. Análisis de los árboles generados

Branch and bound es una técnica que se usa en problemas de optimización para explorar el espacio de soluciones mediante la generación de un árbol de búsqueda. Usando una técnica como dividir (ramificar) y conquistar (poda).

La ramificación o división hace partición del conjunto completo de soluciones factibles en subconjuntos cada vez más pequeños hasta vencer el problema.

La poda o conquista depende de acuerdo a las restricciones que hay en el problema o si el subconjunto no puede contener al óptimo según el límite.

Branch and bound aplica iterativamente el método simplex en cada subproblema individual dentro del algoritmo de branch and bound.



El mecanismo de Branch and bound es el siguiente:

1. Inicialización: Se comienza en el nodo raíz que representa el estado inicial del problema y se establece un límite superior inicial para la función objetivo, que es el valor de la mejor solución conocida hasta el momento.

   Se crea un nodo raíz que representa el estado inicial del problema que sería

la matriz Cal vacía. Se establece un límite inicial, como un valor infinito o un valor muy alto.

En la imagen se puede ver el círculo superior azul que representa al nodo inicial.

2. Expansión de nodos: Se selecciona un nodo activo del árbol de búsqueda y se generan los hijos usando operaciones válidas en el problema. Cada hijo representa un estado sucesor del nodo padre.

   Se selecciona un nodo que representa un calendario parcialmente construido. Se generan todos los posibles hijos los cuales representarían un calendario con una decisión adicional tomada, como por ejemplo la asignación de un equipo a una fecha específica.

   En la imagen se puede ver como del nodo inicial se crean dos nuevos subproblemas.

3. Evaluación de nodos: Se evalúa cada nodo hijo generado en términos de la función objetivo. Si es una solución factible y mejora el límite superior actual, se actualiza el límite superior y se guarda la solución. Si no es factible o su valor objetivo es peor que el límite superior actual, se descarta y no se explora más.

   Se evalúa cada nodo hijo generado en términos de la distancia total de las giras. Si el nodo hijo es factible y mejora el límite superior actual de la distancia de todas las giras, se actualiza el límite superior y tomamos esta solución y la guardamos como la mejor encontrada hasta el momento. En la imagen los rombos en verde representan soluciones factibles encontradas.

   Si este nodo hijo no es factible o su valor de la distancia total de las giras es peor que límite superior actual, se descarta y no se explora más. En la imagen por medio de los triángulos en color rojo se pueden ver los subproblemas que no contienen soluciones y por esto, han sido podados.

4. Ramificación y poda: Si hay nodos pendientes de explorar, se selecciona el próximo nodo activo según la estrategia de selección que se tengan (el nodo más prometedor). Luego, se repiten los pasos 2 y 3.

5. Terminación: Cuando no quedan nodos por explorar.

4. Análisis y pruebas

Resultado de las pruebas: *Link al resultado de las pruebas.*

Se hicieron todas las pruebas de la batería de pruebas incluidas en el enlace anterior y 5 pruebas propias.

Encontramos que para los problemas en el que el número de equipos era pequeño $n = 4$ obtuvimos mejores soluciones en la mayoría respecto a la de los profesores,

Para los problemas con un número de equipos mayor o igual a $n = 6$ en un tiempo de ejecución prudencial (aproximadamente menos de 30 segundos) consideramos que obtuvimos valores muy cercanos al encontrado por los profesores, no alcanzando los mismos valores o mejores, pero sí cercanos.

Para los problemas con un número de equipos de $n = 20$ no alcanzamos a encontrar al menos una respuesta durante un tiempo prudencial.

Esto indica que nuestro modelo mostró un buen rendimiento en los problemas con un número pequeño o moderado de equipos, pero el modelo presenta dificultades en la resolución de problemas de mayor tamaño y complejidad, por lo que pensamos que es posible que nos haya hecho falta tomar en cuenta algunas restricciones adicionales que podrían optimizar el modelo, o también que necesitáramos de técnicas más avanzadas para lograr un mejor desempeño.

También encontramos en internet que este problema se considera NP-hard, lo que significa que no se ha encontrado un algoritmo eficiente para poderlo resolver en un tiempo polinomial para todos los casos posibles.

El problema es más complejo a un mayor número de equipos, a medida que aumenta el número de equipos, el espacio de búsqueda se vuelve exponencial y encontrar un óptimo se hace más difícil. Esto debido a la naturaleza combinatoria del problema y a que cada equipo debe ser programado para jugar contra todos los demás equipos en un calendario sin conflictos y con todas las restricciones.

5. Enlace al vídeo

Enlace al vídeo de la interfaz gráfica

6. Conclusiones

Se realizó el análisis y modelado del problema de la planificación de calendarios de torneos deportivos, identificando los parámetros del problema, las variables de decisión, las restricciones y la función objetivo, la cual se buscaba minimizar.

El modelo propuesto de programación lineal entera es adecuado para resolver el problema de la planificación de calendarios de torneos deportivos. Cumple con todas las restricciones y busca minimizar la distancia total recorrida por los equipos. Pero podría haberse mejorado con la inclusión de la restricción opcional de no programar partidos de vuelta hasta que se hayan programado todos los partidos de ida para mejorar su desempeño.

Añadimos una restricción adicional para garantizar que al menos la mitad de los equipos tengan una permanencia de tamaño Max, la cual consideramos que mejoró un poco el desempeño, pero aún así el tiempo de ejecución es un factor que se podría mejorar, comparado con los resultados de los profesores.

La eficiencia y utilidad en la práctica depende del número de equipos. Para un número igual o mayor a 20 equipos pueden presentar tiempos de ejecución

que se podrían considerar como muy largos y no útiles en la práctica.

Aprendimos a resolver problemas de optimización utilizando técnicas de programación lineal entera y las técnicas que se usan como el método de branch and bound para resolver el problema de la planificación de calendarios de torneos deportivos.