UTN
UNIVERSIDAD TECNOLÓGICA NACIONAL

UNIVERSIDAD TECNOLÓGICA NACIONAL

INGENIERÍA EN SISTEMAS DE INFORMACIÓN

PROYECTO FINAL

Algoritmos de Fixture

20/07/2014

PROYECTO

Que Golazo!

Sistema de Gestión de Torneos de Fútbol

DOCENTES

Ing. Zohil, Julio
Ing. Liberatori, Marcelo
Ing. Jaime, Natalia

ALUMNOS

Allemand, Facundo leg. 58971
Herrera, Antonio leg. 57824
Pedrosa, Paula Melania leg. 58822
Rojas Amaya, M. Florencia leg. 58577

GRUPO
N°3

CURSO
5K2

HISTORIAL DE VERSIONES			
VERSION	**FECHA**	**RESPONSABLE**	**OBSERVACION**
1.0	20/07/2014	Florencia Rojas	Creación Documento

TABLA DE CONTENIDO

Introducción 2

Área temática 2

Objetivo de la investigación 2

FUNDAMENTOS 2

Planificación Round-robin 3

Aplicación de Round-Robin - Metodología 3

sistema todos contra todos 3

Ejemplo de Implementación de Algoritmo 1

Sistema Todos contra todos agregando condiciones 2

EJEMPLO IMPLEMENTACIÓN 2

Bibliografia 5

INTRODUCCIÓN

ÁREA TEMÁTICA

Algoritmos de resolución de modelos matemáticos. Programación Lineal. Cátedras relacionadas: Investigación Operativa.

OBJETIVO DE LA INVESTIGACIÓN

El objetivo de esta investigación es encontrar el/los algoritmos más eficientes que permitan la generación del fixture de un Campeonato de fútbol. Si bien existen muchos algoritmos sencillos para la generación de fixture "todos contra todos", muchas veces estos no satisfacen las necesidades de los Campeonatos en los que su diagramación está determinada por el cumplimiento o no de una serie de condiciones. Dichos requisitos dependen del torneo en particular del que se trate. En caso de no encontrar los algoritmos que se adapten a las necesidades del sistema que se piensa implementar, se buscará diseñar un algoritmo propio que resuelva el problema planteado.

FUNDAMENTOS

Para un mismo torneo existen muchas formas de armar un fixture. Lo que queremos lograr, es llevar a cabo el desarrollo de una diagramación que se adecue a las distintas necesidades que posean cada uno de los organizadores de torneos de fútbol. Por lo tanto, para que el fixture pueda cumplir con sus expectativas, es necesario que contemple distintas precondiciones que deberían tenerse en cuenta desde el primer momento.

En los torneos amateurs también existen muchas veces condiciones que se deben cumplir. En el caso que se trate de partidos ida y vuelta, se tiene que considerar que en el partido de vuelta el equipo local y visitante deben estar invertidos con respecto a la fecha de ida. Además, se puede considerar el hecho que un equipo no juegue dos fechas seguidas como local.

Para un mismo Campeonato, se pueden obtener muchos fixtures diferentes, pero puede que estos no tengan el mismo valor para el organizador. Por esta razón, se busca poder dar soporte a todos los tipos distintos de diagramaciones y a todas las posibilidades de generación de las mismas para que el cliente pueda elegir la óptima.

Armar el fixture adecuado y deseado para cada torneo dependerá de la eficiencia del algoritmo, y allí radica nuestro desafío.

Estos algoritmos deberán estar basados en modelos matemáticos que permitan resolver problemas de satisfacción de restricciones (modelos no relajados).

Problemas de satisfacción de restricciones (CSPs) por sus siglas en Inglés, son problemas matemáticos definidos como un conjunto de objetos tal que su estado debe satisfacer un número de restricciones o limitaciones. CSPs representa las entidades de un problema como una colección homogénea finita de restricciones sobre variables, las que son resueltas por métodos de satisfacción de restricciones. CSPs son el tema de una intensa investigación en Inteligencia Artificial e Investigación de operaciones, dado que la generalidad en su formulación provee un principio básico para analizar y resolver problemas de distintos tipos. CSPs a menudo muestran gran complejidad, requiriendo una combinación de métodos heurísticos y búsqueda combinatoria para ser resueltos en un tiempo razonable. [1]

PLANIFICACIÓN ROUND-ROBIN

Round robin es un método para seleccionar todos los elementos en un grupo de manera equitativa y en un orden racional, normalmente comenzando por el primer elemento de la lista hasta llegar al último y empezando de nuevo desde el primer elemento. El nombre del algoritmo viene del principio de Round-Robin conocido de otros campos, donde cada persona toma una parte de un algo compartido en cantidades parejas.

APLICACIÓN DE ROUND-ROBIN - METODOLOGÍA

La metodología para programar un fixture para un torneo de futbol bajo el sistema round robin se basa en escoger un equipo comodín que es justamente el que salva la situación hipotética de que a un equipo le tocase enfrentarse a sí mismo, lo cual simple y sencillamente no es posible. El resto de los equipos, sea cual sea su número, debe seguir el rol normal de actividades, es decir el equipo x se enfrenta a un contrincante, el cual debe enfrentar en la siguiente jornada al equipo marcado con el número siguiente, x+1, luego al equipo x+2 y así sucesivamente.

SISTEMA TODOS CONTRA TODOS

Si n es el número de competidores, una ronda simple de este sistema requiere de n(n-1)/2 encuentros. Si n es un número par, entonces en (n-1) rondas, se pueden jugar (n-1) partidos simultáneamente. Si n es impar, habrá n rondas con (n-1)/2 juegos simultáneos y un equipo libre (sin jugar) por cada ronda.

Para explicar como desarrollamos este algoritmo vamos a utilizar un ejemplo de 11 equipos (denotados como [1,2,3,4,5,6,7,8,9,10,11]), los cuales se enfrentaran en una sola ronda, de once fechas, con cinco partidos en cada una, quedando un equipo libre en cada fecha por ser un numero impar de equipos.

La estrategia a utilizar será dejar un equipo como "pivote", de la siguiente manera:

FECHA 1	
1	LIBRE
2	11
3	10
4	9
5	8
6	7

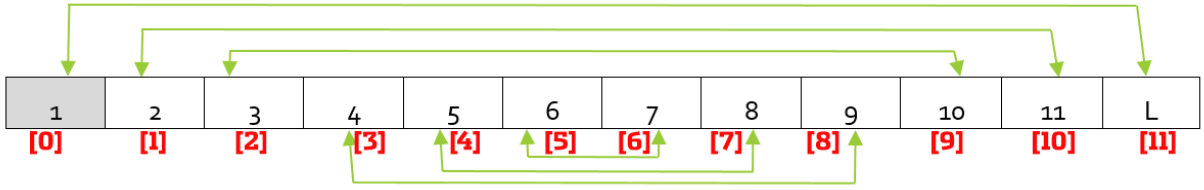
FECHA 2	
1	11
LIBRE	10
2	9
3	8
4	7
5	6

....

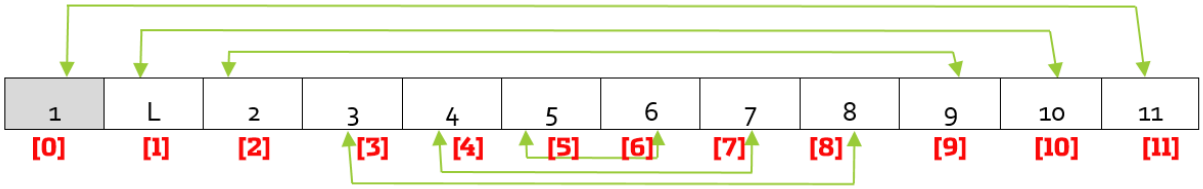
FECHA 11	
1	2
3	LIBRE
4	11
5	10
6	9
7	8

Como puede verse, en cada fecha los equipos van "rotando" en la tabla en sentido antihorario, quedando el equipo "pivote" fijo en su posicion.
Esto puede observarse de la siguiente forma en una lista:

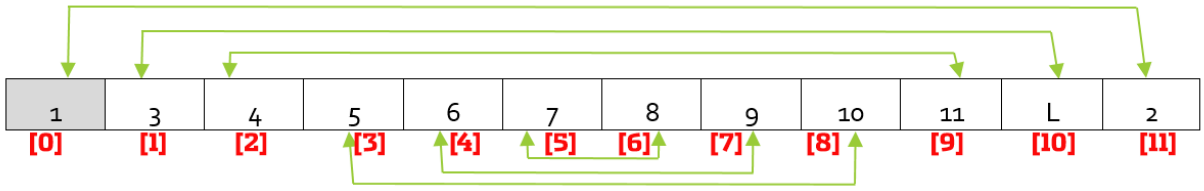
FECHA 1 :



FECHA 2:



FECHA 11:



Puede verse asi el patron de intercambio de posiciones necesario en la lista

para lograr los enfrentamientos entre todos los equipos: se pasa el ultimo elemento de la lista a la posición [1], y desde el elemento que se encontraba en la posicion [1] inclusive se mueven todos los elementos una posicion a la derecha.

EJEMPLO DE IMPLEMENTACIÓN DE ALGORITMO

A continuación se muestra un ejemplo de algoritmo Round Robin implementado para un torneo con 10 equipos:

1ª ronda: (A contra J, B contra I, ...)

A B C D E

J I H G F

2ª ronda: (A contra I, J contra H, ...)

A J B C D

I H G F E

```
/**
*
* @author
*/
public class RoundRobin {

   public static void main(String[] args) {
      RoundRobin rr = new RoundRobin();
      for (int i = 0; i < 9; i++) {
         rr.mostrar();
         rr.combinar();
      }
   }

   private String[] equipos= new String[10];

   public RoundRobin() {
      this.equipos[0]="A";
      this.equipos[1]="B";
      this.equipos[2]="C";
      this.equipos[3]="D";
      this.equipos[4]="E";
      this.equipos[5]="F";
      this.equipos[6]="G";
```

```
    this.equipos[7]="H";
    this.equipos[8]="I";
    this.equipos[9]="J";
  }

  public void combinar(){
    String buffer=equipos[equipos.length-1];

    for (int i = equipos.length-1; i > 1; i--) {
      equipos[i]=equipos[i-1];
    }
    equipos[1]=buffer;
  }

  public void mostrar(){
    for (int i = 0, j=equipos.length-1; i<j; i++, j--) {
      System.out.println(equipos[i]+" vs "+ equipos[j]);
    }
    System.out.println("**************************************");
  }
}
```

SISTEMA TODOS CONTRA TODOS AGREGANDO CONDICIONES

A continuación se presenta otro ejemplo de implementación, en el que se consideran algunas restricciónes más, como es el hecho de considerar la localía de los equipos y que es de tipo "TODOS CONTRA TODOS IDA Y VUELTA".

EJEMPLO IMPLEMENTACIÓN

El problema que se resolverá es el siguiente:

Es necesario diseñar un fixture para organizar los partidos de un campeonato de fútbol. Las restricciones que tiene el problema son las siguientes:

- Existen 4 equipos que deben jugar.
- Deben jugar todos contra todos.
- Hay dos partidos por cada par de equipos posibles (intercambiando los roles de local y visita).
- Un equipo sólo puede jugar un partido por fecha.
- Dos equipos no pueden jugar entre sí en 2 fechas consecutivas, es decir, si River Plate juega con Boca en la fecha 5, en la fecha 6 estos dos equipos no pueden enfrentarse nuevamente.
- Un equipo no puede jugar más de 2 veces consecutivas de local o de visita, es decir, si River jugó sus últimos dos partidos de local, en el próximo partido debe jugar de visita.

Deducciones:

- Cantidad de equipos = a = 4
- Cantidad de partidos = b
- Cantidad partidos simultáneos = c
- Cantidad de fechas = d

La cantidad de partidos a jugar se puede calcular mediante la variación entre la cantidad de equipos que participarán en el campeonato y la cantidad de equipos que participan por partido, esto es la variación entre 4 y 2.

$$b = \frac{a!}{(a-2)!} = \frac{4!}{(4-2)!} = \frac{24}{2} = 12$$

La cantidad de partidos simultáneos por fecha corresponde a la mitad de la cantidad de equipos que participarán en el campeonato.

$$c = \frac{a}{2} = \frac{4}{2} = 2$$

Finalmente, la cantidad de fechas corresponde al cociente entre la cantidad de partidos y la cantidad de partidos simultáneos.

$$d = \frac{b}{c} = \frac{12}{2} = 6$$

A continuación se describe un algoritmo en pseudo-código aplicando este algoritmo:

```
Arreglo[][] algoritmoFixture(Arreglo[4] equipos){
    CONSTANTE entero numeroFechas = 6;
    CONSTANTE entero numeroEquipos = 4;
    CONSTANTE entero numeroPartidos = 12;
    Arreglo[numeroPartidos][3] fixture;
    int local,visita;
    int programandoPartido = 1;
    int programandoFecha = 1;
    Arreglo[numeroEquipos] disponibles;
    Arreglo[numeroEquipos][ numeroEquipos] matriz ;
    Arreglo[numeroEquipos][2] registroHap;
```

```
      Arreglo[numeroEquipos] ultimaFecha;
      boolean iguales, disponibilidad, programado, rival, hapLocal, hapVisita,  hap ;
      llenarMatriz(matriz);
      llenarHap(registroHap);
      while(programandoFecha<=numeroFechas){
        reiniciarDisponibles(disponibles);
        Para local=0 hasta numeroEquipos-1 {
          Para visita=0 hasta numeroEquipos-1{
            iguales = (local != visita);
            programado = matriz[local][visita]==0;
            disponibilidad = disponibles[local]==0 && disponibles[visita]==0;
            rival = visita != ultimaFecha[local];
            Si ( registroHap[local][0]!=0 || (registroHap[local][0]==0
&&
                 registroHap[local][1]>0) )
                        hapLocal = true;
                 Sino
                        hapLocal = false;
                Si ( registroHap[visita][0]!=1 || (registroHap[visita][0]==1 &&
                     registroHap[visita][1]>0) )
                        hapVisita = true;
                 Sino
                        hapVisita = false;
                        hap = hapLocal && hapVisita;
            Si (iguales && disponibilidad && programado && rival && hap){
              matriz[local][visita]=programandoPartido;
             fixture[programandoPartido-1][0] = tring.valueOf(programandoFecha);
              fixture[programandoPartido-1][1] = equipos[local];
              fixture[programandoPartido-1][2] = equipos[visita];
              disponibles[local]=1;
              disponibles[visita]=1;
              ultimaFecha[local] = visita;
              ultimaFecha[visita] = local;
              Si (registroHap[local][0]==0)
                registroHap[local][1]--;
              Sino{
                registroHap[local][0] = 0;
                registroHap[local][1] = 1;
              }
              Si (registroHap[visita][0]==1)
                registroHap[visita][1]--;
              Sino{
                registroHap[visita][0] = 1;
```

```
                registroHap[visita][1] = 1;
              }
              programandoPartido++;
            }
          }
        }
        programandoFecha++;
      }
      retornar(fixture);
    }
```

BIBLIOGRAFIA

http://es.wikipedia.org/wiki/Problema_de_satisfacci%C3%B3n_de_restricciones

http://www.algovidea.cl/index.php?option=com_content&view=article&id=72&Itemid=84

http://es.wikipedia.org/wiki/Sistema_de_todos_contra_todos

http://es.wikipedia.org/wiki/Planificaci%C3%B3n_Round-robin

http://isc-lemg.blogspot.com.ar/2012/09/hoy-traigo-el-algoritmo-de-round-robin.html