Monografia do projeto do Curso de Introdução ao Escalonamento
Geração de tabelas de futebol

Rodrigo L. M. Flores
#USP 5127470

13 de dezembro de 2009

1 Introdução

Torneios de futebol há muito tempo deixaram de ser apenas mero entretenimento para se tornar espetáculos televisivos, com audiências em níveis bastante elevados e torcidas ocupando cada cadeira do estádio. Estes espetáculos movimentam alguns bilhões de reais no mundo, seja em patrocínios, transferências de jogadores, arrecadação da venda de ingressos, venda de produtos licenciados por clubes, entre outras coisas.

Torcedores se tornam apaixonados pelos seus clubes e a rivalidade com torcedores de outro clube é tão grande a ponto de os confrontos com torcedores de uma torcida adversária formarem um ambiente semelhante ao de uma guerra, vandalizando o ambiente em sua volta e deixando torcedores mortos e feridos.

Embora este problema seja bem comum no Brasil, em outros países como Argentina, Inglaterra, Escócia, Turquia e Itália também costumam ter problemas parecidos podendo (ou não) ter um plano de fundo político, étnico ou religioso. Por exemplo, os times *Rangers* e *Celtic* da cidade de *Glasgow*, Escócia acabam por incluir nesta briga o ódio já existente entre protestantes e católicos.

A questão televisiva também é importante: normalmente não é permitido que a televisão aberta transmita um jogo para a cidade na qual acontecerá o jogo. Isso incentiva torcedores a irem ao estádio e acompanharem ao vivo as partidas do seu time. Exceções costumam acontecer em jogos finais nos quais os estádios normalmente estão lotados e o jogo tem muita audiência.

Sabendo disso, é necessário gerar tabelas de torneio que levem em conta esse aspecto de não permitir que dois times da mesma cidade joguem no mesmo dia, na mesma cidade. Como algumas cidades, como São Paulo e Rio de Janeiro, há mais de dois times, é necessário que as rodadas sejam divididas em pelo menos dois dias. Se chamarmos o número máximo de equipes de uma mesma cidade de M, o número de dias que uma rodada deve durar deve ser $\lceil \frac{M}{2} \rceil$. Felizmente, o número de times de uma mesma cidade dificilmente passa de 4, o que permite que rodadas durem apenas 2 dias (um fim de semana ou uma quarta e quinta).

O objetivo deste trabalho é: dado times e suas cidades, gerar uma tabela para um campeonato de pontos corridos (isto é, todos os times se enfrentando em jogos de ida e volta) sem que dois times joguem na mesma cidade em uma mesma rodada em um mesmo dia, ou, especificamente, sem que mais de 2 times joguem como mandante na mesma cidade na mesma rodada (considerando uma rodada durando 2 dias).

2 Possíveis abordagens

Uma das possíveis abordagens para a solução deste problema é gerar todas as tabelas possíveis e buscar as que atendem ao nosso critério. Como o número de combinações para 20 equipes (da ordem de 10^{130}) é gigantesco, podemos rapidamente descartar esta solução. Uma outra abordagem seria gerar um padrão de jogos e fazer uma busca nestas soluções: gerar todas as combinações é da ordem de $n!$, que para n pequeno (menor que 25) é bem possível fazer este cálculo.

Como muitos times não tem esse problema por eles não terem outros times da mesma cidade disputando o mesmo campeonato, podemos pensar em outra abordagem: resolver primeiro os times que têm esse problema, e as vagas que sobrarem serão sorteadas para os outros times. Assim garantimos uma solução mais rápida.

3 Algoritmo de geração de padrões

O algoritmo, retirado do artigo [Sch90], gera os padrões de jogos (chamados no artigo de *HAP*, *Home Away Pattern*) da seguinte maneira: dado um jogo gerador (que é o jogo do time i com o $i+1$ na rodada i) ele gera os outros jogos da rodada da seguinte maneira:

Algorithm 1 Algoritmo para geração das rodadas

$invert \leftarrow TRUE$
$round \leftarrow [generatorGame]$
for $i = 1$ to $\lfloor N/2 \rfloor$ **do**
  $team1 \leftarrow normalize(generatorGame.home + i)$
  $team2 \leftarrow normalize(generatorGame.away - i)$
  if $invert$ **then**
    $round.append((team2, team1))$
    $invert \leftarrow FALSE$
  else
    $round.append((team1, team2))$
    $invert \leftarrow TRUE$
  end if
end for

A rotina $normalize$ apenas tira o módulo N e adiciona 1 de modo a sempre termos um número no intervalo 1 a N. O 1 resolve o problema para um número ímpar de times, já para um número par precisamos fazer uma artimanha: resolvemos para um número $N-1$ de times e o time que sobrou joga com o time N, que

alterna entre jogos em casa e jogos fora de casa.

Uma quebra de padrão *Home-Away* é quando um time joga duas partidas seguidas fora ou dentro de casa. Para um número ímpar de times, o algoritmo tem 0 quebras de padrão *Home-Away*, já para um número par de times acontecem $2N - 2$ quebras, o que pode não ser bom mas ainda assim é aceitável. A prova desta propriedade pode ser vista no artigo [Sch90].

Um campeonato sem restrição seria uma atribuição de cada time a um destes padrões. Porém, para o nosso problema, queremos atribuir de modo a minimizar o número de jogos no mesmo dia na mesma rodada.

3.1 Propriedade especial do algoritmo *HAP*

Para $N = 20$, descobriu-se uma propriedade bastante interessante: utilizando um dos pares dentre $(1, 2)$,$(3, 4)$,$(5, 6)$,$(7, 8)$,$(9, 10)$,$(11, 20)$,$(12, 13)$,$(14, 15)$,$(16, 17)$ e $(18, 19)$, podemos colocar dois times da mesma cidade atribuidos aos pares e sempre que um deles jogar em casa o outro jogará fora de casa. Isso permite que utilizemos estes pares para colocar dois times da mesma cidade e eles não terão jogos simultaneamente dentro ou fora de casa (como o campeonato tem turno e returno, os dois jogando fora de casa certamente terá os dois jogando dentro de casa no returno). Chamaremos esses pares de *pares complementares*.

3.2 Atribuição dos times

A atribuição dos times aos padrões é feito da seguinte maneira:

1. Obtemos uma lista das cidades que tem mais de um time. Guardamos também em uma lista o complemento desta.

2. Utilizando a lista das cidades que tem mais de um time, atribuimos, por meio de um sorteio, para cada par de times um par complementar. Se o número de times da cidade for ímpar, o time restante é incluído na lista de cidades com apenas um time.

3. Atribuimos para cada atribuição de padrões um time sorteado de alguma das cidades pertencente à lista de cidades com apenas um time.

Ao fim deste processo, temos o campeonato gerado, sem mais de dois times jogando na mesma cidade na mesma rodada, o que é o melhor caso da minimização proposta. Embora o programa utilize um algoritmo combinatório para encontrar os pares, podemos colocá-los *hardcoded* no programa, para assim resolvê-lo em um tempo não exponencial.

4 Implementação

A implementação foi feita na linguagem de programação Ruby. O código foi dividido em 4 pastas:

- **bin -** Contém o gerador de tabelas propriamente dito. Considere este programa como o *main* deste script;
- **data -** Contém algumas entradas que podem ser utilizadas para a utilização do programa;
- **lib -** Contém as implementaçõs de HAP, da classe de times, do parser do arquivo de entrada. Há também um arquivo;
- **spec -** Contém os testes do gerador de *HAP*, do parser e da classe de times. que define algums métodos adicionais criados para a classe *Array* que facilitaram o trabalho.

Detalhes de como se executar podem ser encontrados no arquivo README.md na pasta raíz do projeto.

5 Conclusão

O objetivo proposto na introdução foi cumprido: temos uma tabela com nenhum jogo na mesma cidade no mesmo dia. Porém há duas sugestões de melhorias: evitar que uma equipe enfrente dois adversários fortes ou fracos em seguida, minimizar a distância percorrida por cada time (embora isto seja pouco utilizado na prática, pois os times normalmente voltam para sua cidade para os jogadores irem para suas casas ou para enfrentarem times em outros campeonatos como copa do Brasil, que pode ter uma partida em qualquer estado do Brasil ou a taça libertadores, que pode ter uma partida em qualquer país da América do Sul ou no México).

Referências

[Sch90] Jan A. M. Schreuder, *Combinatorial aspects of construction of competition dutch professional football leagues*, Discrete Applied Mathematics **35** (1990), 301–312.