PAA – Relatório do Trabalho Prático 3

Danilo Ferreira e Silva[1]

[1]Departamento de Ciência da Computação – UFMG

danilofs@dcc.ufmg.br

1. Introdução

Neste trabalho foi projetado um algoritmo para resolver o problema de encontrar times matematicamente eliminados de um campeonato, ou seja, que não possuem mais chances de ficar em primeiro lugar (ou empatado em pontos). Para tal, o algoritmo analisa o máximo de pontos que um time pode alcançar comparado aos seus adversários, considerando todos os jogos que ainda serão disputados. Um exemplo de entrada é dado na tabela 1.

Tabela 1. Exemplo de entrada do problema

Time	Vitórias	Derrotas	A jogar	vs. A	vs. B	vs. C	vs. D
A	83	71	8	0	1	6	1
B	80	79	3	1	0	0	2
C	78	78	6	6	0	0	0
D	77	82	3	1	2	0	0

2. Modelagem da solução

Seja $T = \{t_1, t_2, ..., t_n\}$ o conjunto de n times da divisão. Seja t_x o time de interesse que deseja-se verificar a possibilidade matemática. O conjunto dos concorrentes a t_x, ou seja, os times de T com exceção de t_x, será chamado de T^x.

Como estamos interessados na existência de alguma chance de t_x não ser superado por outro time, podemos considerar as seguintes condições como verdadeiras, pois elas sempre levam ao melhor cenário para t_x:

- t_x ganhará todas as partidas que restam a ele;
- os demais times perderão todas as partidas que não sejam contra times da mesma divisão.

Com isso limitamos nossa análise apenas para o conjunto de jogos J que sejam entre times $i, j \in R$. Todos estes jogos terão algum vencedor, e estes pontos serão distribuídos a algum time dentro do próprio conjunto T^x.

O problema pode ser modelado como uma rede de fluxo onde desejamos distribuir $|J|$ vitórias para os times T^x. No entanto, desejamos restringir essa distribuição de pontos de tal forma que nenhum time supere t_x.

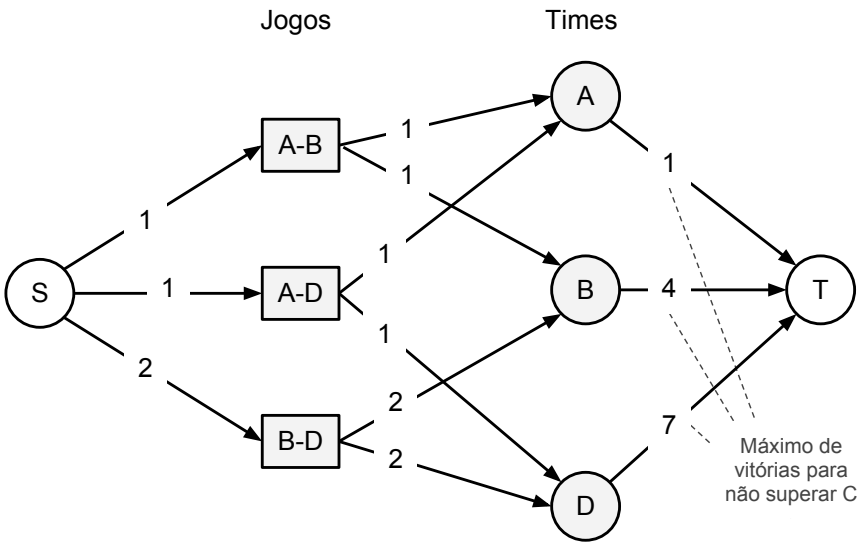

Figura 1. Problema da tabela 1 modelado como uma rede de fluxos

No figura 1 temos o grafo da rede de fluxos associada ao problema dado na tabela 1, onde analisamos as chances do time C. Partindo do vértice S temos arestas que ligam a cada partida que deverá ser disputada, onde a capacidade da aresta é o número de partidas. Saindo de cada partida temos duas arestas que levam aos times que a disputam. A princípio qualquer um dois dois times pode ganhar todas as partidas, portanto a capacidade de ambas as arestas é a quantidade de jogos disputados.

Finalmente, saindo de cada time temos uma única aresta ligada ao destino T. A capacidade destas arestas deve ser atribuída de forma que o time não possa ganhar mais partidas que t_x. No exemplo dado, C atinge no máximo 84 vitórias. Portanto B, que já possui 80, não pode ganhar mais do que 4 partidas.

O time analisado está eliminado se, e somente se, o fluxo máximo da rede for menor que $|J|$. Se o fluxo for igual a $|J|$ existe uma atribuição de resultados a todas as partidas de J em que t_x não é eliminado.

2.1. Justificativa para a eliminação

Outra informação que podemos obter da rede de fluxos no caso de eliminação é o conjunto de times R que terão um número de vitórias agregadas tal que algum deles garantidamente superará t_x.

Seja e_i a aresta que liga um time t_i ao destino na rede de fluxos. Se a capacidade de e_i for maior que o fluxo atribuído a mesma, podemos dizer que as partidas que envolvem t_i não tiveram como vencedores times que pudessem superar t_x. Caso isso ocorresse, seria possível obter um fluxo maior mudando o vencedor dessa partida para t_i, o que é uma contradição.

Desta forma, eliminando cada time t_i que tenha capacidade excedente em sua aresta de saída e os jogos que envolvem t_i da rede, ainda assim teremos uma rede com fluxo máximo menor que o fluxo que sai da fonte (partidas a jogar), e todos os times

estarão empatados com t_x. Logo, temos o conjunto de times R que explicam a eliminação de t_x.

3. Análise de complexidade

Como discutido acima, verificar a eliminação de um time t_x envolve a construção de uma rede de fluxos e encontrar o fluxo máximo da mesma. Isso foi feito utilizando o algoritmo Edmonds-Karp, uma variante do Ford-Fulkerson que usa a busca em largura para encontrar caminhos de aumento. A complexidade do mesmo é $O(VE^2)$.

Dada uma entrada com n times, o quantidade máxima de confrontos distintos entre eles é $(n-1)^2/2$. Portanto, cada em cada grafo construído temos:

$$V = 1 + (n-2)^2/2 + (n-1) + 1 = (1/2)n^2 + n - 1 = O(n^2)$$
$$E = 3(n-2)^2/2 + (n-1) = (3/2)n^2 + n - 7 = O(n^2)$$

Portanto, o custo total no pior caso é:

$$\begin{aligned} C &= nO(VE^2) \\ &= nO(n^2(n^2)^2) \\ &= O(n^7) \end{aligned}$$

Na figura 2 temos o gráfico do tempo de execução pelo tamanho da entrada, expresso pelo número de times n. Nele podemos observar o crescimento rápido da curva, que é condizente com a complexidade $O(n^7)$.

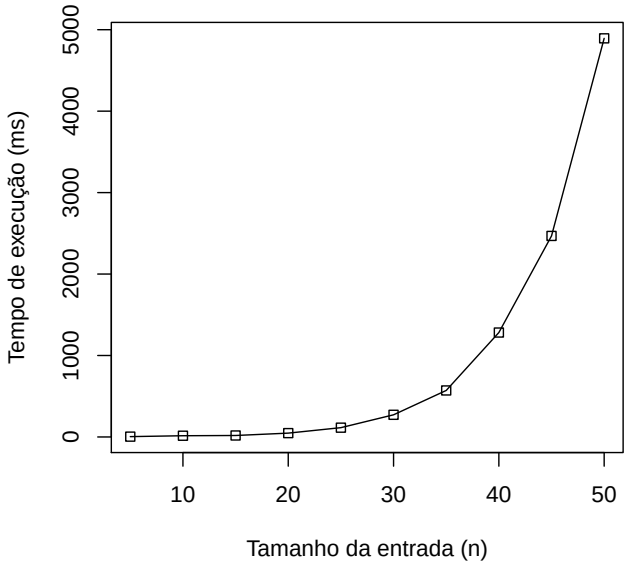

Figura 2. Tempo de execução por número de times n

Em termos de espaço, o custo é apenas $O(n^2)$, tanto para armazenar a entrada quando o grafo necessário.

4. Conclusão

Neste trabalho foi desenvolvido um algoritmo para detectar times eliminados em um campeonato. Para tal o problema foi modelado em um grafo que expressa uma rede de fluxos. O problema de verificar se um time estava eliminado foi então reduzido em encontrar o fluxo máximo deste grafo.

A complexidade final do algoritmo foi $O(n^7)$. Em experimentos foi constatado que o crescimento do tempo de execução de fato é muito elevado. Embora o algoritmo seja polinomial, ele começa a se tornar inviável para entradas cujo o número de times seja mais do que algumas dezenas.