Documentação TP3 - Atualizações Problemáticas

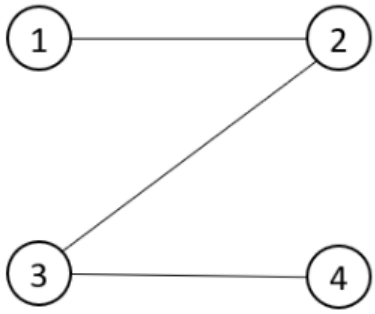
Bruno M. Monteiro

30/06/2018

1 Introdução

Este trabalho teve como objetivo a resolução do seguinte problema: dada uma rede de servidores, quer-se atualizar todos eles. Porém, há a restrição de que dois servidores adjacentes não podem ser atualizados juntos, pois o sistema ficaria, então *offline*.

Portanto, seriam feitas várias rodadas de atualização, de forma que em cada rodada dois servidores adjacentes não sejam atualizados. Por exemplo, se houverem quatro servidores e o servidor 2 for adjacente ao 1 e ao 3, e o servidor 3 for também adjacente ao quatro, uma possível solução seria atualizar na primeira rodada os servidores 1 e 3, e na segunda rodada atualizar os servidores 2 e 4. O exemplo está representado no seguinte grafo:



Dado o número de servidores e as relações entre os servidores, o problema era encontrar uma forma de atualizar a rede com o mínimo de rodadas possíveis. Deve-se encontrar o número mínimo de rodadas e quais servidores serão atualizados em cada rodada.

2 Solução do Problema

O problema foi modelado inicialmente em um grafo, onde os vértices são os servidores e há uma aresta uv se u e v são adjacentes.

Deve-se então notar que podemos designar para cada vértice uma cor, de forma que se todos os vértices de uma dada cor serão atualizados na mesma rodada. O problema se torna então COLORAÇÃO. Mais especificamente, encontrar o número cromático do grafo, um problema conhecidamente NP-Completo.

Um algoritmo força-bruta foi feito, que testa se um gafo pode ser colorido usando k cores. Testa-se então qual o menor $k \geq 1$ que tem como resposta SIM. O algoritmo testa todas as possibilidades de cor (representada por um inteiro de 0 a $k-1$) para o vértice i, e repete para o vértice $i+1$. Quando $i = n$, testamos se a coloração é válida. Para o seguinte algoritmo, $f(1)$ nos responde se um grafo G é k-colorível.

Algorithm 1 Força-bruta

Input: grafo G, inteiro k
Output: SIM, se G é k-colorível; NÃO, caso contrário.

```
1: function f(i)
2:     if i = |V(G)| + 1 then
3:         if ∃u, v ∈ V(G) | cor[u] = cor[v] then return NÃO
4:         else return SIM
5:     else
6:         for todo 1 ≤ i ≤ |V(G)| do
7:             if f(i + 1) then return SIM
           return NÃO
```

Em seguida, foi feita uma heurística, dado que o algoritmo força-bruta é exponencial. A heurística funciona da seguinte forma: para cada vértice, seleciona-se para a cor dele a menor cor que não foi selecionada para seus vizinhos. Olha-se então a maior cor utilizada, e esse valor é a aproximação do número cromático de grafo.

Nota-se que a ordem de avaliação dos vértices interfere na resposta encontrada. Portanto, o algoritmo heurística é executado várias vezes, com ordenações aleatórias de vértices, e vê qual é a melhor solução obtida.

Algorithm 2 Heurística

Input: grafo G
Output: k, aproximação do número cromático de G.

```
1: for todo vértice i do
2:     S ← ∅
3:     for todo vizinho j de i do
4:         if j possui cor then
5:             S ← S ∪ {cor[j]}
6:     for c de 1 a |V(G)| do
7:         if c ∉ S then
8:             cor[i] ← c
9:             vá para o próximo vértice
       return maior cor atribuída
```

3 Análise de Complexidade Assintótica

3.1 Complexidade Assintótica de Tempo

O algoritmo de força bruta tenta colorir com k cores. Para tentar colorir com k cores, ele tenta todas as possibilidades, isto é, para cada vértice, tenta todas as cores. Pode-se ver claramente que há k^n possibilidades de coloração.

Para cada possibilidade de coloração, o algoritmo verifica se ela é válida, e para isso olha se para par de vértices adjacentes, eles possuem a mesma cor. Isso é feito em $\mathcal{O}(n^2)$.

Observa-se que todo grafo G pode ser colorido com $|V(G)|$ cores, atribuindo para cada vértice sua própria cor. Ou seja, $1 \leq k \leq n$ Assim, temos que no pior caso o algoritmo leva $\mathcal{O}(n^n)\mathcal{O}(n^2) = \mathcal{O}(n^{n+2})$.

Já a heurística, por outro lado, é um algoritmo polinomial. Para cada vértice, visita-se todos os seus vizinhos e olha qual a menor cor que os vizinhos não usam. Isso é feito em $\mathcal{O}(n)$. Portanto, a heurística possui complexidade $\mathcal{O}(n^2)$.

3.2 Complexidade Assintótica de Espaço

Tanto no algoritmo força-bruta como na heurística, a maior estrutura de dados é a lista de adjacência utilizada para armazenar o grafo. Portanto, a complexidade de espaço é, nos dois casos, $\mathcal{O}(n + m)$.

4 Análise Experimental

Para fazer a analise experimental. foram gerados grafos bipartidos completos, de forma que o algoritmo força-bruta levaria tempo $\mathcal{O}(2^n)$. Dessa forma, pudemos observar as curvas esperadas na análise teórica. Os gráficos resultantes se encontram abaixo.

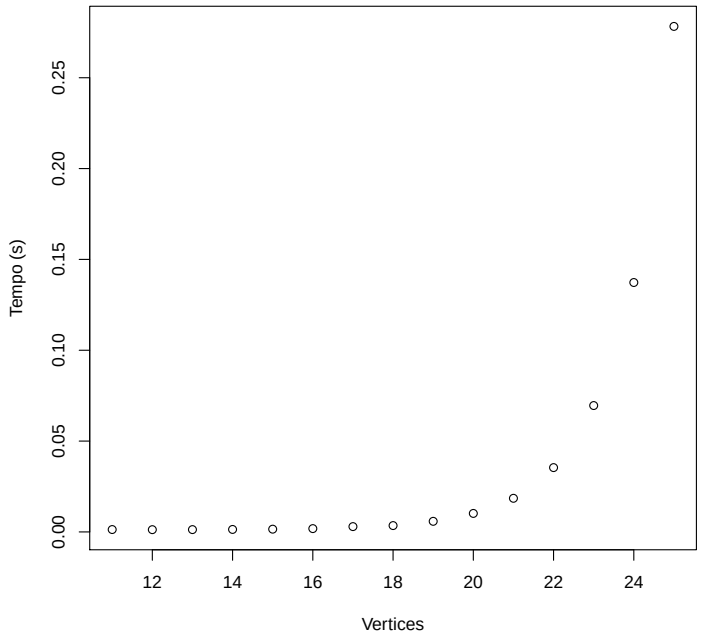


Figure 1: Tempo de execução do algoritmo força-bruta

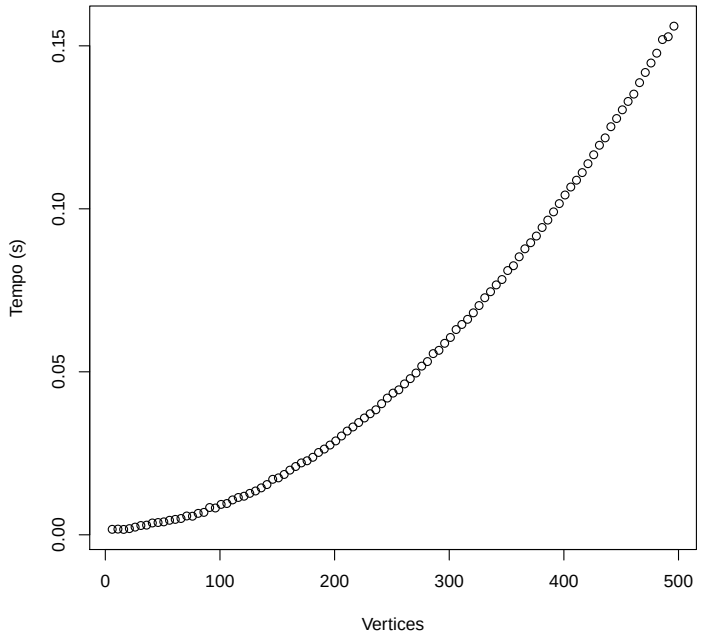


Figure 2: Tempo de execução da heurística

As saídas da heurística para os exemplos estão a seguir. Na primeira linha, vê-se o resultado correto. Na segunda, o resultado da heurística.

4, 5, 6, 8, 10, 13, 30, 5, 31, 54
4, 5, 6, 8, 12, 18, 30, 14, 31, 54

5 Conclusão

Neste trabalho, pode-se analisar na prática a dificuldade de tratamento de um problema NP-Completo. Algoritmos exponenciais foram desenvolvidos e analisados experimentalmente.

Pudemos estudar o funcionamento de heurísticas, e de como podemos, na prática, tentar aproximar ao máximo da solução correta.

Os gráficos obtidos experimentalmente confirmam nossas análises teóricas, sendo o referente ao algoritmo força-bruta exponencial, e o da heurística, quadrático.

Observou-se que a heurística, embora acerte muitas vezes, pode ser bem pior que a saída ótima. Entretanto, seu tempo de execução é incomparavelmente mais rápido que o algoritmo força-bruta.