Documentação - Trabalho Pratico 3 - AEDS III

Aluno – Jesimon Barreto Santos (2016070093)

1 Introdução

Você acaba de ser contratado para ser o novo administrador de redes da empresa BH Software, criadores de um aplicativo de comunicação por áudio e vídeo que vem ganhando popularidade. O aplicativo é suportado por uma rede de servidores que ocasionalmente precisam ser desconectados para receberem novas atualizações de segurança. O problema que você se depara é que realizar a atualização de um a um de todos os servidores demoraria muito, já atualizar todos os servidores ao mesmo tempo, é inviável pois o aplicativo ficaria offline. Além disso, um servidor e seus adjacentes na rede não podem ser atualizados simultaneamente pois o tráfego do servidor offline é redirecionado para seus vizinhos.

Uma solução é alocar os servidores para serem atualizados em rodadas diferentes, obedecendo as restrições impostas. O seu objetivo como administrador da rede é propor algoritmos que descubram o número mínimo de rodadas e a alocação dos servidores necessárias para que seja feito a atualização no menor tempo possível. Por exemplo, dado uma configuração de rede simples mostrada na figura 1, uma possível forma de realizar as atualizações seria em duas rodadas, onde na primeira, os servidores 1 e 3 seriam atualizados, depois os servidores 2 e 4 já que não possuem conexões entre si.

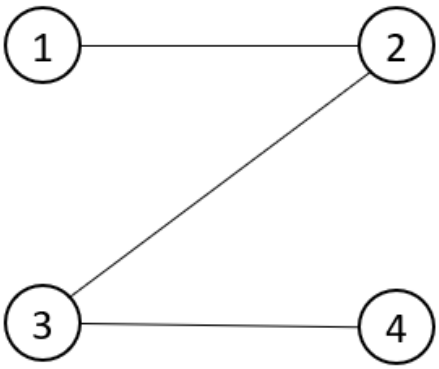

Figura 1. Rede de Servidores

2 Abordagem do Problema

Com base no problema apresentado, foi induzida uma abordagem baseado no algoritmo de Força Bruta e uma heurística escolhida foi a gulosa para colorir grafo. Algoritmo usado para calcular o conjunto de servidores que podem atualizar ao mesmo tempo, para o força bruta, é necessário calcular todo o espaço de soluções (todas as possíveis soluções), até encontrar a ótima. No caso da heurística gulosa, a tomada de decisão baseia-se apenas no próximo passo.

Dado um grafo com n vértices, a força bruta analisa, a partir de k grupos igual a 1, até a quantidade mínima necessária, permutando todos os n vértices, descartando aqueles conjuntos

não válidos para a solução. Naturalmente, a primeira solução encontrada será a ótima (mínima quantidade), pois o número de conjuntos começa com o menor possível e vai aumentando. O algoritmo de Força Bruta analisa toda a arvore de recursão do conjunto solução do problema.

No caso da Heurística Gulosa, dado um grafo com n vértices a preocupação é cumprir a regra da solução analisando apenas o próximo passo. Nesse caso, verificado os vértices um a um, se ainda não possui conjunto definido, é visto se os vértices adjacentes já possuem conjunto definido, se sim é salvo e é dado o menor valor de conjunto para o vértice v que seja diferente dos seus adjacentes.

Algoritmo guloso geral para resolver esse problema é dado por [3]:

Enquanto houver vértice incolor faça

seja v um vértice incolor

se alguma cor i não é usada por nenhum vizinho de v

então atribua cor i a v

senão atribua cor k a v e faça k = k+1

Representação usada foi lista de adjacência, pois o grafo é não orientado e pode possuir muitas arestas.

3 Complexidade

A complexidade do programa é explicada em termos do algoritmo, pois é a principal implementação desse trabalho prático. A implementação do algoritmo de força bruta não foi implementado.

Iniciando com a explicação da complexidade de tempo, começando pelo algoritmo de força bruta:

$$O(k^n)$$

Onde:

k é o número de conjuntos mínimos necessário para regra imposta

n é o número de vértices no grafo

Para verificar o espaço:

$$E\ (N * M)$$

Onde:

N é o número de vértices.

M é o número de arestas**.**

A complexidade de tempo do algoritmo de heurística gulosa, no pior caso:

$$O(N*[N-1])$$

Onde:

N é o número de vértices no grafo

Se explica porque o grafo pode ser esparso e assim, pode ser que cada vértice esteja conectado com os outros N-1 vértices e como explicado, o algoritmo verifica todos os vértices e a cada vértice analisa seus vizinhos.

Para verificar a complexidade de espaço, usando lista de adjacência temos que:

$$E\ (N * M)$$

Onde:

N é o número de vertices.

M é o número de arestas**.**

4 Experimentos

Os testes foram feitos em um computador com processador i5, 2.6 GHz e memória de 1 GB, no sistema operacional Ubuntu. Os casos de teste disponibilizados foram executados com sucesso, seguem o comportamento expresso como a seguir:

Tabela1. Tempo em segundos necessário para executar cada arquivo de teste heurística.

Arquivo	tempo(s)	Saída	Arquivo Rodada
Entrada1	0.00001	4	4
Entrada2	0.00001	5	5
Entrada3	0.00004	6	6
Entrada4	0.00004	8	8
Entrada5	0.00009	16	10
Entrada6	0.00053	21	13
Entrada7	0.00214	30	30
Entrada8	0.00246	18	5
Entrada9	0.00345	31	31
Entrada10	0.00693	54	54

Como foi mostrado no tópico de Abordagem do problema, o algoritmo geral mostra que são encontrados valores menores de conjunto para o mesmo grafo que é proposto o arquivo resultado. A heurística resolve bem o problema de coloração de grafo em um tempo muito curto. O algoritmo foi rodado nos arquivos de força bruta para comparar o resultado, e o obtido foi o ótimo, em tempo menor que 0.0 segundos para cada um dos casos de força bruta.

Tabela2. Tempo em segundos necessário para executar cada arquivo de teste heurística nos arquivos de força bruta.

Arquivo	Saída	Rodada
Entrada1	2	2
Entrada2	3	3
Entrada3	2	2
Entrada4	4	4
Entrada5	3	3
Entrada6	3	3
Entrada7	2	2
Entrada8	3	3
Entrada9	7	7
Entrada10	8	8
Entrada11	9	9
Entrada12	10	10
Entrada13	20	20
Entrada14	3	3
Entrada15	3	3
Entrada16	30	30
Entrada17	2	4
Entrada18	40	40
Entrada19	3	3
Entrada20	60	60
Entrada21	100	100
Entrada22	2	2
Entrada23	200	200
Entrada24	400	400
Entrada25	1000	1000
Entrada26	3	3

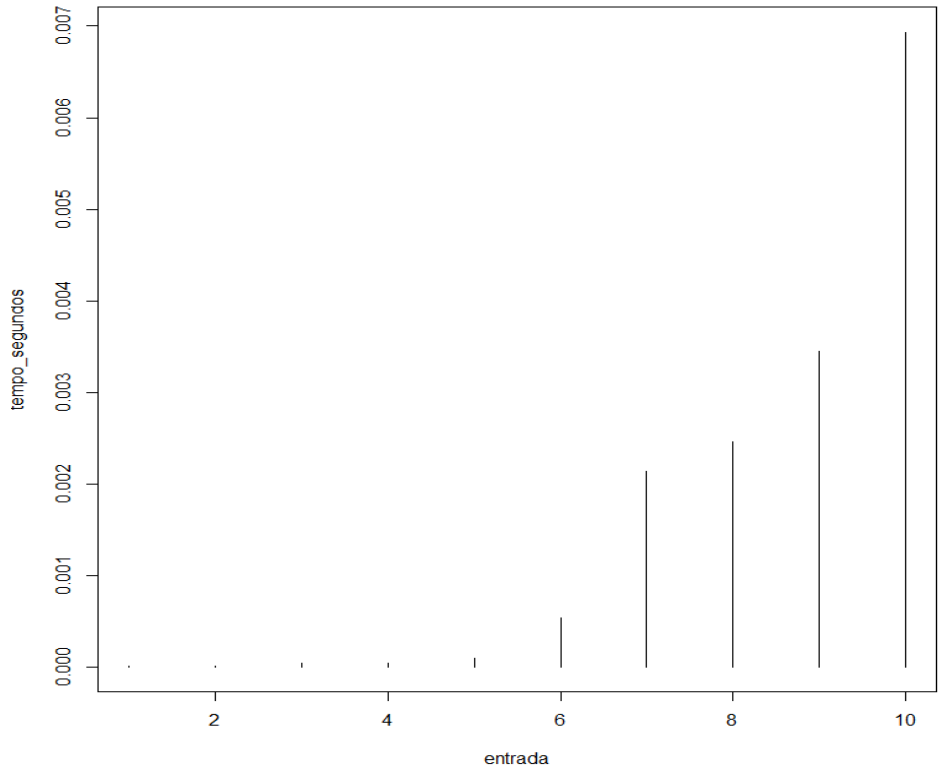

Figura1. Gráfico que relaciona tempo em segundos com entradas de teste de Heurísticas referente a tabela1.

O gráfico exibe, em formato de histograma, o resultado de tempo por entrada. O eixo x é o input e o y é o tempo em segundos da execução de cada entrada.

5 Conclusão

Esse programa foi criado para resolver um problema de coloração de vértices em um grafo, cujo, é dado um grafo e a regra é que dois vizinhos não podem ter a mesma cor, ou pertencer ao mesmo conjunto, e é necessário encontrar a menor quantidade de cor possível e a combinação. Para resolver esse problema foi usado um algoritmo heurística gulosa.

Referencial Teórico

1. Ziviani, N. *Projeto de Algoritmos com Implementações em Pascal e C*, São Paulo, Brazil, Cengage Learning, ISBN 13 978-85-221-1050-6, 2010, third edition reviewed and extended, 659 pages (in Portuguese).
2. André Backes. *Linguagem C: Completa e Descomplicada*. Editora Campus Elsevier, 2012.
3. https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/vertex-coloring.html
4. https://www.geeksforgeeks.org/graph-coloring-set-2-greedy-algorithm/