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.

Como podemos perceber, o problema é idêntico ao de coloração de grafos. Como esse último é um problema NP-Completo clássico, vou sempre me referir ao problema original como se fosse esse.

Para o restante do trabalho, serão usadas as seguintes notações: $V =$ total de vértices; $m =$ número de cores; $|v| =$ índice do vértice.

2. Solução do Problema

Na modelagem do problema, os grafos foram representados através de uma matriz de adjacência, já que a operação de verificar se existe uma aresta de $u \rightarrow v$ é utilizada constantemente e tem complexidade de $O(1)$ nessa estrutura.

Para a solução por força bruta, o algoritmo implementado foi um de busca por força bruta simples, que gera todas as possíveis colorações e verifica se alguma delas é solução para a coloração do grafo utilizando m cores. Para descobrir o número mínimo de cores necessário para resolver o problema, a força bruta é rodada de maneira incremental em relação ao parâmetro m até que exista uma solução.

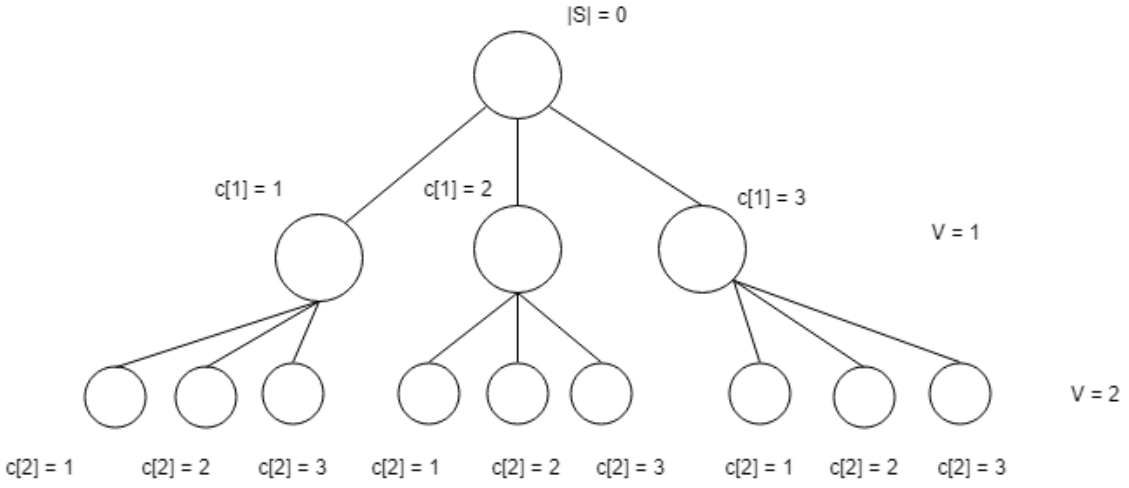


Figura 1: Exemplo de uma árvore de possíveis colorações encontrada pelo algoritmo de força bruta em um grafo com $V = 2$ e $m = 3$.

Para a solução heurística, foi implementado um algoritmo guloso que colore sequencialmente os vértices ordenados do grafo. Primeiro, o algoritmo colore o vértice 1, depois, para o restante dos vértices, ele escolhe para um vértice v a menor cor possível que ainda não tenha sido usada por algum vizinho de v. Se todas as cores já tiverem sido escolhidas, é criada uma nova cor.

Entretanto, como já foi mencionado, a solução gulosa resulta em uma heurística, pois ela não garante um m mínimo, já que a subestrutura do problema, que depende de uma escolha arbitrária em relação a ordem dos vértices a serem coloridos, não é ótima.

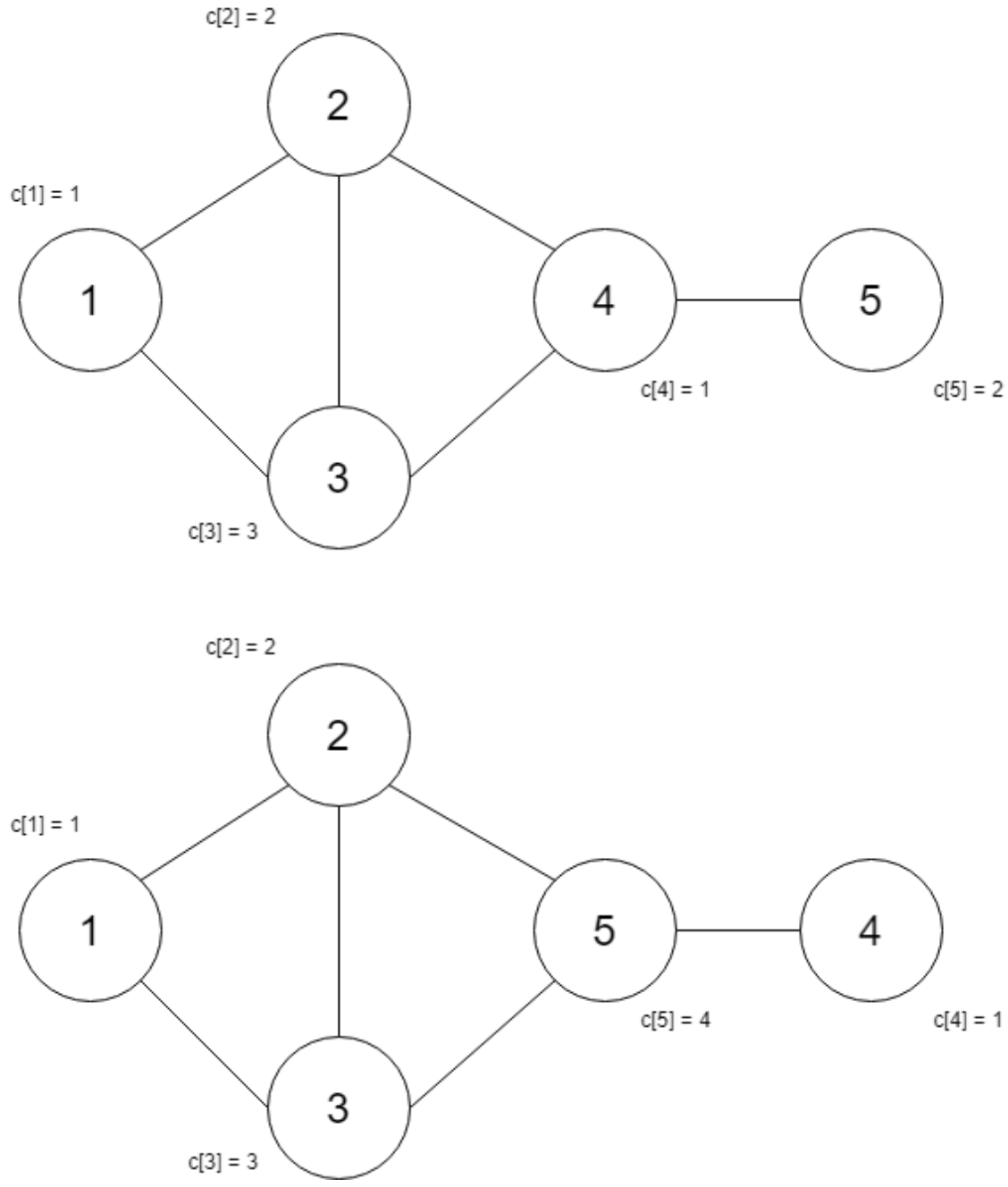


Figura 2: Exemplo que demonstra a heurística do algoritmo guloso.

Como podemos ver, pela solução heurística, o primeiro grafo pode ser colorido com 3 cores, enquanto o de baixo pode ser colorido com 4 cores.

3. Análise de Complexidade

O algoritmo de força bruta testa todas as colorações possíveis com m cores até que se encontre uma solução. Como cada vértice do grafo pode ser colorido de m maneiras diferentes, temos que o total de combinações é igual a m^V. Portanto, a complexidade total do algoritmo é de $O(m^V)$

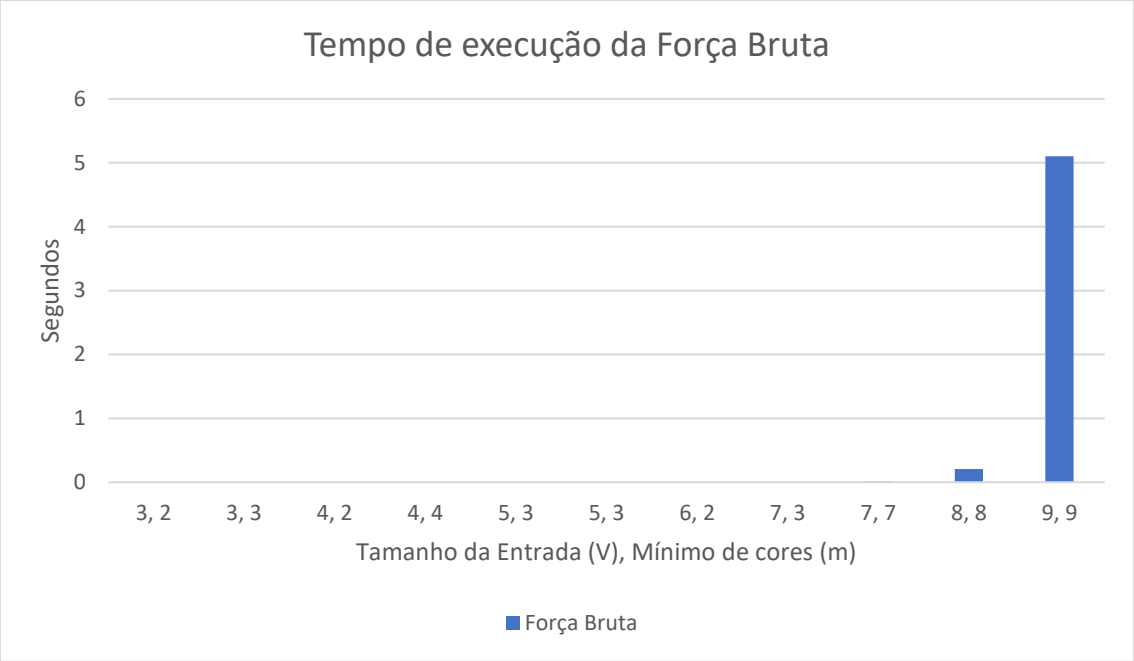
Já a solução heurística colore cada vértice com a menor cor que seja compatível com seus vizinhos. Se todos as cores já foram escolhidas, é criada uma nova cor. Cada vértice é comparado com todos os seus vizinhos e são testadas até m configurações de cores para cada vértice. Portanto, sua complexidade é de $O(m \cdot V^2)$

Como mencionado anteriormente, para ambas as soluções foram utilizadas matrizes de adjacência para representar os grafos, portanto a complexidade de espaço para ambos é de $O(V^2)$.

4. Análise Experimental

Para a análise experimental, foram utilizados os testes fornecidos e o tempo de execução é dado pela média de 10 execuções da função de coloração.

Tamanho da entrada (V)	Força Bruta	Heurística	Mínimo de rodadas (F.B)	Mínimo de rodadas (H)	Diferença entre H – F.B
3	0.000000s	0.000000s	2	2	0
3	0.000001s	0.000000s	3	3	0
4	0.000001s	0.000000s	2	2	0
4	0.000005s	0.000000s	4	4	0
5	0.000003s	0.000000s	3	3	0
5	0.000004s	0.000000s	3	3	0
6	0.000002s	0.000000s	2	2	0
7	0.000014s	0.000001s	3	3	0
7	0.009210s	0.000001s	7	7	0
8	0.206259s	0.000001s	8	8	0
9	5.101847s	0.000001s	9	9	0
10	Não rodou	0.000001s	-	10	-
20	-	0.000005s	-	20	-
21	-	0.000002s	-	3	-
25	-	0.000002s	-	3	-
30	-	0.000015s	-	30	-
40	-	0.000002s	-	4	-
40	-	0.000031s	-	40	-
55	-	0.000006s	-	3	-
60	-	0.000094s	-	60	-
100	-	0.000383s	-	100	-
202	-	0.000049s	-	2	-
200	-	0.002232s	-	200	-
400	-	0.017029s	-	400	-
1000	-	0.264215s	-	1000	-
1105	-	0.001202s	-	3	-



Já para a heurística, todos os tempos de execução foram menores que 0.27 segundos, então um gráfico não ajudaria muito bem a entender o crescimento do tempo.

5. Conclusão

Como podemos perceber na análise experimental, o algoritmo de força bruta realmente possui um tempo exponencial de complexidade.

Já a solução heurística, mesmo com complexidade quadrática, demonstra um tempo de execução extremamente superior ao de força bruta e, mesmo não garantindo um m mínimo, conseguiu chegar em uma solução ótima para todos os testes que rodaram na força bruta. Além disso, também percebe-se que os piores casos da heurística ocorrem quando $m = V$, ou seja, quando todos os vértices possuem cores diferentes. Isso faz sentido, pois para cada iteração da função recursiva, uma nova cor é criada, ou seja, para cada v, vão ser testadas $|v|$ configurações de cores.