UNIVERSIDADE FEDERAL DE JUIZ DE FORA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
DCC059 - TEORIA DOS GRAFOS 2018-3

CRISTIANO NASCIMENTO DA SILVA
DIOGO FREITAS DE ANDRADE
ELENO TRINDADE GOMES JUNIOR
JOÃO VICTOR LOPES FAM
MATHEUS ESTEVÃO FARIA

TRABALHO DE TEORIA DOS GRAFOS

COLORAÇÃO DE VÉRTICES APLICADA EM REDES SEM FIO

RELATÓRIO

JUIZ DE FORA

Novembro - 2018

CRISTIANO NASCIMENTO DA SILVA
DIOGO FREITAS DE ANDRADE
ELENO TRINDADE GOMES JUNIOR
JOÃO VICTOR LOPES FAM
MATHEUS ESTEVÃO FARIA

TRABALHO DE TEORIA DOS GRAFOS

COLORAÇÃO DE VÉRTICES APLICADA EM REDES SEM FIO

Relatório da segunda parte (final) do trabalho que serve como parte das atividades avaliativas previstas na disciplina de Teoria dos Grafos (DCC059) do Departamento de Ciência da Computação.

Professor: Stênio Sã Rosário Furtado Soares

JUIZ DE FORA

Novembro - 2018

SUMÁRIO

1 INTRODUÇÃO 3
2 METODOLOGIA UTILIZADA 4
2.1 ESTRUTURA DE DADOS UTILIZADAS 4
2.2 ABORDAGENS ALGORÍTMICAS USADAS NA SOLUÇÃO 5
2.2.1 Algoritmo Guloso 5
2.2.2 Algoritmo Guloso Randomizado 6
2.2.3 Algoritmo Guloso Randomizado Reativo 6
3 EXPERIMENTOS COMPUTACIONAIS 8
4 CONCLUSÃO 10
REFERÊNCIAS 11

1 INTRODUÇÃO

Redes de transmissão de dados sem fio são essenciais atualmente. Sinais de rádio e de celular são exemplos de sinais transmitidos ininterruptamente por toda a sociedade. Visto que é necessário cobrir a maior área possível com estes sinais a fim de garantir o melhor uso pela população, se torna inevitável que o raio do sinal fornecido por diferentes torres se encontrem. Para resolver este problema, torres próximas emitem sinais em frequências diferentes, reduzindo assim a interferência entre elas e proporcionando uma experiência mais agradável para os usuários. Porém, apenas isto não basta para garantir a não-interferência completa entre os sinais das torres, pois as faixas de frequência operáveis são um recurso escasso, e cada uma possui uma certa interferência com cada uma das outras. Sendo assim, a proposta deste trabalho é justamente alocar frequências de trabalho para cada torre, minimizando a interferência entre cada interseção.

2 METODOLOGIA UTILIZADA

O problema foi modelado em grafos como um grafo G(V,E) onde V é o conjunto de torres pertencentes ao problema e E é o conjunto de relações entre as torres (arestas) onde existe relação se uma torre causa interferência com a outra. O objetivo do problema é alocar uma frequência a cada vértice do grafo de tal forma que minimize o somatório das interferências. Este problema foi baseado no artigo *"Abordagens heuristicas para o problema de atribuicao de potencia e de canais de transmissao em redes cognitiva"*(FERREIRA, 2015).

No desenvolvimento deste trabalho ultilizamos a linguagem de programação C++ para representar e resolver o problema. O grafo foi representado utilizando lista de adjacência, sendo que cada nó possui uma propriedade referente à frequência alocada para este.

2.1 ESTRUTURA DE DADOS UTILIZADAS

Para desenvolver os algoritmos gulosos, utilizamos como base o TAD GrafoListaAdj já desenvolvido na parte anterior do trabalho, porém, ao invés de permanecer com o uso da lista encadeada fornecida pela biblioteca "list" do C++, nós utilizamos do "vector" fornecido pelo próprio C++, de forma que o grafo é um vector de nós e cada nó possui um vector com os adjacentes, sendo mais específico para a resolução do nosso problema. No que diz respeito à classe Vertice, foi adicionado o atributo "frequência" para armazenar a frequência com a qual cada vértice (torre de transmissão) opera.

Utilizamos de uma matriz de interferência representada por um vetor com valores do tipo float que armazena o valor de interferência entre os canais x e y na posição (x, y) da matriz, que é calculada pela equação $((x-1) \times 14) + (y-1))$, sendo que o "$-1$" é porque os canais vão de $1-14$ enquanto a matriz $0-13$. A matriz foi preenchida de acordo com os valores obtidos na Figura 1, e a lista de candidatos é inicializada com a própria lista de vértices do grafo.

Matriz de sobreposição														
Canal	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	1	0.77	0.54	0.31	0.09	0	0	0	0	0	0	0	0	0
2	0.77	1	0.77	0.54	0.31	0.09	0	0	0	0	0	0	0	0
3	0.54	0.77	1	0.77	0.54	0.31	0.09	0	0	0	0	0	0	0
4	0.31	0.54	0.77	1	0.77	0.54	0.31	0.09	0	0	0	0	0	0
5	0.09	0.31	0.54	0.77	1	0.77	0.54	0.31	0.09	0	0	0	0	0
6	0	0.09	0.31	0.54	0.77	1	0.77	0.54	0.31	0.09	0	0	0	0
7	0	0	0.09	0.31	0.54	0.77	1	0.77	0.54	0.31	0.09	0	0	0
8	0	0	0	0.09	0.31	0.54	0.77	1	0.77	0.54	0.31	0.09	0	0
9	0	0	0	0	0.09	0.31	0.54	0.77	1	0.77	0.54	0.31	0.09	0
10	0	0	0	0	0	0.09	0.31	0.54	0.77	1	0.75	0.5	0.31	0
11	0	0	0	0	0	0	0.09	0.31	0.54	0.77	1	0.75	0.54	0
12	0	0	0	0	0	0	0	0.09	0.31	0.54	0.77	1	0.77	0.22
13	0	0	0	0	0	0	0	0	0.09	0.31	0.54	0.77	1	0.45
14	0	0	0	0	0	0	0	0	0	0	0	0.22	0.45	1

Figura 1: Tabela de interferências

2.2 ABORDAGENS ALGORÍTMICAS USADAS NA SOLUÇÃO

2.2.1 ALGORITMO GULOSO

O algoritmo guloso desenvolvido pode ser definido pelas seguintes informações:

1. Lista de Candidatos (LC): vector do tipo Vertice que armazena as informações dos vértices dos grafos, há dois atributos para a identificação do vértice e o atributo "grau" que não necessariamente é sempre igual ao grau do vértice que representa. A Lista de Candidatos, entretanto, é inicializada refletindo os dados de cada vértice do grafo.

2. Função critério: a função $f : LC \rightarrow R$ dada por $f(x) = x.\text{grau}$ é a função critério utilizada no algoritmo, sendo aplicada para ordenar a Lista de Candidatos a cada iteração de maneira decrescente.

3. Atualização da LC: o algoritmo de atualização da Lista de Candidatos remove o elemento da lista na posição solicitada (no caso do algoritmo guloso, tal posição é sempre 0).

Para concluir a descrição do guloso falta mostrar como é gerada a solução do problema. A cada ordenação da Lista de Candidatos, é selecionado o primeiro elemento com maior grau e aplicado a função *selecionaFrequencia(Vertice* vertice)*, que adiciona uma frequência ao vértice passado por parâmetro. Após isso, tal elemento é removido

da lista com a atualização da lista de candidatos, e este processo é repetido enquanto a LC possuir algum elemento. Ao fim do algoritmo, o grafo vai estar com as frequências devidamente definidas, e pode-se inclusive retornar a interferência total da rede como critério de comparação.

A função *selecionaFrequencia* busca encontrar que frequência pode ser atribuída ao vértice minimizando a interferência total com os adjacentes que já tiveram frequência definida (a frequência 0 indica que ela ainda não foi definida, e o adjacente é desconsiderado). Para isto, ele calcula a interferência total obtida com cada uma das 14 bandas de frequência e seleciona a banda que gerou menor interferência. Apesar da abordagem se assemelhar à força bruta, o caráter local (minimiza apenas a interferência com adjacentes) e o número constante de bandas de frequência a serem testadas tornam a função executável em um tempo polinomial que se encaixa na abordagem gulosa.

2.2.2 ALGORITMO GULOSO RANDOMIZADO

A forma randomizada adiciona dois parâmetros à chamada da função, que antes não possuía parâmetros em sua assinatura. A influência desses parâmetros no algoritmo é o que dá o caráter randomizado do algoritmo guloso. São os parâmetros:

1. itTotal (número de iterações): É o valor que define a quantidade de iterações que serão realizadas antes do algoritmo retornar o melhor resultado. Esse parâmetro só faz sentido no algoritmo guloso quando adicionamos a ideia de randomização, pois no guloso não há mudança do resultado em diferentes iterações. Dessa forma, cada iteração pode ter como resultado um valor de interferência total diferente, e o melhor valor será salvo durante o processo.

2. alfaR: É o coeficiente que deve assumir um valor entre 0 e 1. Define até qual ponto da Lista de Candidatos que o algoritmo considera na hora de escolher um vértice aleatoriamente. Ou seja, o índice do vértice na LC que terá a frequência definida (e que será removido da lista depois) vai ser escolhido em um intervalo entre 0 e alfa$\times|LC|$, onde $|LC|$ é o tamanho da Lista de Candidatos.

2.2.3 ALGORITMO GULOSO RANDOMIZADO REATIVO

A abordagem gulosa randomizada reativa difere da anterior apenas na seleção do valor de alfa. Ao invés de selecionar um valor fixo para alfa, um valor aleatório é selecionado a cada iteração. Os valores de alfa que produzirem os melhores resultados

têm mais chances de serem selecionados novamente. Inicialmente os valores entre 0 e 1 são separados em uma quantidade fixa de números igualmente espaçados que serão os possíveis alfas. Para cada possível alfa é armazenado a soma de todas as soluções obtidas com aquele alfa e o número de vezes que ele foi utilizado.

As probabilidades de cada alfa ser selecionado são igualmente distribuídas inicialmente. Após um dado número de iterações (gamma) é calculado um peso para cada alfa baseado em quão próximo da melhor solução ficaram os resultados desse alfa. A probabilidade de cada alfa é então distribuída de acordo com esses pesos e o "betaRR" define quando vai ocorrer a mudança de um alfa para outro.

3 EXPERIMENTOS COMPUTACIONAIS

Os testes de desempenho do algoritmo foram realizados numa máquina com as seguintes especificações:

1. CPU: i5 5200U, Dual Core, 2.20 GHz
2. RAM: 8GB, DDR3, 1600 MHz
3. Sistema Operacional: Linux MINT 18.2 Sonya

Todas as instâncias foram executadas 30 vezes para cada heurística.

Instâncias	Melhor	Guloso	Rand 0.1	Rand 0.2	Rand 0.3	Reativo	Melhor α
instance_3	0	0	0	0	0	0	0.6
instance_5	0.93	1.68	1.68	1.68	1.68	0.93	0.6
instance_7	0	0	0	0	0	0	0.2
instance_11	54.31	68.69	58.65	57.17	56.67	54.31	0.2
instance_13	52.81	65.33	60.39	56.62	54.48	52.81	0.2
instance_17	55.45	68.74	62.22	60.76	60.52	55.45	0.8
instance_19	56.7	69.76	61.66	62.72	60.84	56.7	1
instance_23	216.32	252.06	239.61	233.54	227.41	216.32	1
instance_29	201.91	234.74	219.69	216.38	214.08	201.91	0.6
instance_30	198.86	233.77	222.19	216.62	208.27	198.86	0.4

Tabela 1: Melhores Soluções

Instâncias	Randomizado 0.1	Randomizado 0.2	Randomizado 0.3	Reativo
instance_3	0	0	0	0
instance_5	1.68	1.68	1.68	1.68
instance_7	0	0	0	0
instance_11	59.9337	58.688	58.205	55.635
instance_13	61.3493	58.7917	57.1097	54.983
instance_17	64.0997	62.938	62.153	57.3377
instance_19	65.2253	64.192	62.7317	58.694
instance_23	242.782	238.693	232.942	219.819
instance_29	225.855	221.389	217.612	205.989
instance_30	225.37	220.908	214.926	203.609

Tabela 2: Média dos resultados

Instâncias	Randomizado 0.1	Randomizado 0.2	Randomizado 0.3	Reativo
instance_3	0	0	0	0
instance_5	1.68	1.68	1.68	0
instance_7	0	0	0	0
instance_11	0.0741854	0.0551385	0.256517	0.241911
instance_13	0.0656042	0.192006	0.0566574	0.396732
instance_17	0.0329233	0.230408	0.0918352	0.0890354
instance_19	0.0922607	0.0971297	0.0489908	0.0956679
instance_23	0.257728	0.163764	0.109969	0.0713933
instance_29	0.150565	0.611747	0.0897688	0.293763
instance_30	0.432703	0.374645	0.690921	0.476641

Tabela 3: Desvio Padrão

4 CONCLUSÃO

Em suma, o problema em questão é classificado como NP completo, e é computacionalmente inviável buscar a solução ótima, portanto utilizamos abordagens heurísticas a fim de obter soluções aproximadas. Para isso, utilizamos três métodos construtivos: Algoritmo Guloso, Guloso Randomizado e Guloso Randomizado Reativo. Analisando os dados coletados, é possível concluir que abordagem puramente gulosa é a mais ingênua dentre as três, visto que apenas busca os melhores candidatos localmente, e nem sempre isso traz a melhor solução possível. Diferente disso, as abordagens gulosa randomizada e gulosa randomizada reativa procuram os melhores candidatos utilizando um fator de aleatoriedade e compara diversas soluções diferentes permitindo que o algoritmo se "arrependa" de soluções intermediárias encontradas e melhorando a qualidade da solução final.

REFERÊNCIAS

FEOFILOFF, P. **Algoritmos para Grafos em C via Sedgewick**. 2017. Disponível em: <www.ime.usp.br/∼pf/algoritmos_para_grafos/>. Acesso em: 01 de outubro de 2017.

FEOFILOFF, P. **Grafos bipartidos não-dirigidos e circuitos ímpares**. 2017. Disponível em: <https://www.ime.usp.br/∼pf/algoritmos_para_grafos/aulas/bipartite.html>. Acesso em: 17 de outubro de 2017.

FERREIRA, L. R. Abordagens heuristicas para o problema de atribuicao de potencia e de canais de transmissao em redes cognitiva. 2015.

NETTO, P. O. B. **Grafos: teoria, modelos, algoritmos**. [S.l.]: Edgard Blücher, 2003.

SZWARCFITER, J. L. Grafos e algoritmos computacionais.[sl]: Editora campus ltda. **Rio de Janeiro-1984**, 1984.