Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação
Teoria dos Grafos

Algoritmos Gulosos para o Problema do Subconjunto Dominante Mínimo

Grupo 8

Anthony Lima e Silva – **MAT** 201765255AC
Daniel Machado Barbosa Delgado – **MAT** 201835013
Gabriel Bronte Cardoso – **MAT** 201835002
Giovane Nilmer de Oliveira Santos – **MAT** 201835012
Guilherme Marques de Oliveira – **MAT** 201835008
Marcos Mateus Oliveira dos Santos – **MAT** 201835019

Professor: Stênio Sã Rosário F. Soares

Relatório do trabalho final da disciplina DCC059 - Teoria dos Grafos, parte integrante da avaliação da mesma.

Juiz de Fora

Novembro de 2020

1 Introdução

Foi escolhido o problema do Conjunto Dominante Mínimo para um dado grafo. É um problema de decisão NP-completo em teoria de complexidade computacional – ou seja, não existe um algoritmo que consiga encontrar o Conjunto Dominante Mínimo de um grafo em tempo polinomial.

A solução escolhida foi o uso dos algoritmos guloso e guloso randomizado, propostos em sala de aula.

2 Descrição do problema

O problema do Conjunto Dominante Mínimo consiste em encontrar um subconjunto de vértices em um dado grafo G, onde todo vértice de G ou está nesse conjunto, ou é adjacente a um vértice presente nele. Tal subconjunto deve possuir o menor número possível de vértices.

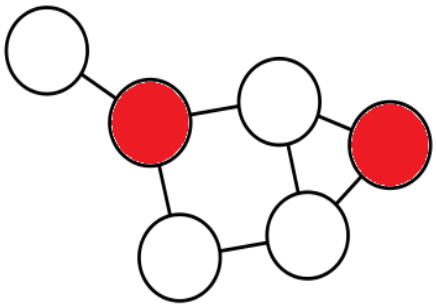


Figura 1: Exemplo de subconjunto dominante mínimo em um grafo, com seus membros pintados de vermelho.

Problemas NP-difíceis de grande relevância de Conjunto Dominante Mínimo estão em aplicações relacionadas a projetos de rede sem fio, mineração de dados, teoria de códigos, entre outros. Um exemplo mais ligado ao cotidiano seria encontrar, em um jogo de xadrez, o menor número de rainhas que dominam todo o tabuleiro, como pode ser visto neste trabalho.

3 Abordagens gulosas para o problema

A heurística do algoritmo foi escolhida com base nos diálogos com o professor e porque foi a melhor compreendida pelo grupo. Já a decisão da estratégia da lista de candidatos se deu pois tal ordenação engloba todos os vértices do grafo, e todos são percorridos até que se encontre o Conjunto Dominante Mínimo.

3.1 Algoritmo guloso

Primeira coisa a ser feita é ordenar uma lista de candidatos com todos os vértices do grafo de maior a menor grau pela função *Quicksort*. Há também uma lista de graus, ordenada paralelamente à de candidatos.

Dentro de uma estrutura de repetição que roda enquanto ainda há candidatos **e** não se encontrou um subconjunto dominante, a função inicia na primeira posição da lista de candidatos, removendo seu nó e adicionando-o a uma outra lista de solução. É também removido na lista de graus o seu grau. Diminui-se também um grau de cada vértice adjacente ao removido, e caso o grau de algum deles chegue a 0, este é removido das listas de candidatos e de graus.

Percorre-se então a lista de graus e verifica se todos os seus membros foram modificados. Se sim, um conjunto dominante foi encontrado. Nesse caso, o algoritmo guloso é encerrado; caso contrário, a lista de candidatos é reordenada com os nós restantes, e o algoritmo repete o processo acima.

3.2 Algoritmo guloso randomizado

A heurística deste algoritmo é a mesma do Guloso normal, porém há um conjunto de alfas (0.1, 0.2, 0.3, 0.5 e 0.7) que randomizam a posição a ser escolhida na lista de candidatos, fazendo a operação (Tamanho da lista de candidatos – 1) * alfa.

Para cada valor de alfa o algoritmo será chamado dez vezes, e em cada uma há um número máximo de 500 iterações realizadas. Se 200 iterações se passarem desde que um inteiro com a solução melhor foi atualizado, o algoritmo para e retorna tal solução.

4 Experimentos computacionais

4.1 Descrição das instâncias

As instâncias foram passadas pelo professor, e foram utilizadas todas:

Instância	Nº de Vértices	Nº de Arestas
Problem.dat_50_50_0	50	50
Problem.dat_50_100_0	50	100
Problem.dat_50_500_0	50	500
Problem.dat_100_250_0	100	250
Problem.dat_100_2000_0	100	2000
Problem.dat_300_300_0	300	300
Problem.dat_300_1000_0	300	1000
Problem.dat_500_1000_0	500	1000
Problem.dat_800_0_0	800	800
Problem.dat_50_5000_0	50	5000

4.1 Ambiente computacional do experimento e conjunto de parâmetros

Para testar os algoritmos foram utilizadas duas máquinas:

- Sistema Operacional Windows 10 – Processador Intel i5 4440, 8 GB RAM;
- Sistema Operacional Windows 10 – Processador Intel i5 7400, 16 GB RAM;

Os valores de alfa utilizados foram (0.1), (0.2), (0.3), (0.5) e (0.7). Para cada alfa, o algoritmo é chamado 10 vezes com no máximo 500 iterações – caso não se encontre melhor solução em 200 iterações.

4.2 Resultados quanto à qualidade e o tempo

Média de Resultados

A média do número de vértices do subconjunto dominante encontrado no guloso e em cada uma das 10 execuções do randomizado para cada valor alfa passado como parâmetro.

As colunas da tabela abaixo indicam, respectivamente, a instância utilizada, o resultado (fixo) do algoritmo guloso, e a média dos resultados de cada valor alfa no guloso randomizado. Ao final, o melhor valor entre eles.

Instâncias	**Guloso**	**Alfa (0.1)**	**Alfa (0.2)**	**Alfa (0.3)**	**Alfa (0.5)**	**Alfa (0.7)**	**Melhor valor**
dat_50_50	23	26.4	29.8	30.6	30.2	27.3	23
dat_50_100	17	22.7	23.1	22.7	23.3	23.2	17
dat_50_500	6	5.1	4.1	4	4	4.6	4
dat_100_250	35	45.4	45.6	43.4	44.6	44.9	35
dat_100_2000	7	5.4	5.3	5.8	5.5	5.9	5.3
dat_300_300	126	238.2	224.5	213.5	202.9	180.8	126
dat_300_1000	121	158	148.6	150.9	153.6	153.9	121
dat_500_1000	249	322.6	312.1	308.2	318.2	318.8	249
dat_800_0	452	663.3	608.6	562	526.8	477.1	452
dat_800_5000	250	312.5	314.3	322.5	316.2	321.5	250

Tempo Médio de Execução

Tempo médio de execução do algoritmo guloso randomizado nas dez iterações de cada alfa.

Instâncias	**Alfa (0.1)**	**Alfa (0.2)**	**Alfa (0.3)**	**Alfa (0.5)**	**Alfa (0.7)**
dat_50_50	0.0409	0.0374	0.0412	0.0385	0.0304
dat_50_100	0.0394	0.0479	0.0368	0.0479	0.0411
dat_50_500	0.0564	0.0487	0.0427	0.0419	0.0405
dat_100_250	0.2213	0.2256	0.1927	0.2094	0.2711
dat_100_2000	0.1648	0.1703	0.1446	0.1701	0.1413
dat_300_300	3.1989	2.7497	3.4152	2.5274	2.491
dat_300_1000	2.5413	2.8031	2.863	2.353	3.0776
dat_500_1000	6.8982	7.3225	7.1136	7.4033	8.0045
dat_800_0	47.4115	45.0115	43.9734	45.3874	40.3201
dat_800_5000	34.5436	28.7404	31.7602	37.5538	37.8585

Desvio Percentual do Tempo Médio de Execução (em segundos)

Demonstra (em %) o desvio percentual do tempo médio gasto para a execução de cada algoritmo com base nas instâncias. as seguintes exibem o desvio das 10 execuções de cada valor alfa no algoritmo guloso randomizado. Não foi inserido a coluna do algoritmo guloso pois ele é executado apenas uma vez.

Instâncias	**Alfa (0.1)**	**Alfa (0.2)**	**Alfa (0.3)**	**Alfa (0.5)**	**Alfa (0.7)**
dat_50_50	34.5	23.0	35.5	26.6	0.0
dat_50_100	7.0	30.1	0.0	30.1	11.7
dat_50_500	39.2	20.2	5.4	3.4	0.0
dat_100_250	14.8	17.0	0.0	8.6	40.6
dat_100_2000	16.6	20.5	2.3	20.3	0.0
dat_300_300	28.4	10.3	37.1	1.4	0.0
dat_300_1000	7.9	19	21.6	0.0	30.7
dat_500_1000	0.0	6.1	3.1	7.3	16.0
dat_800_0	17.5	11.6	9.0	12.5	0.0
dat_800_5000	20.1	0.0	10.5	30.6	31.7

Desvio Percentual da Média dos Resultados

Desvio percentual calculado entre as médias dos resultados do guloso e de cada uma das 10 execuções de cada alfa do guloso randomizado e comparadas com o melhor resultado entre elas. o valor (em %) está em cada coluna seguinte.

Instâncias	**Guloso**	**Alfa (0.1)**	**Alfa (0.2)**	**Alfa (0.3)**	**Alfa (0.5)**	**Alfa (0.7)**
dat_50_50	0	14.7	29.5	33.0	31.3	18.6
dat_50_100	0	33.5	35.8	33.5	37.0	36.4
dat_50_500	50.0	27.5	2.5	0	0	15.0
dat_100_250	0	29.7	30.2	24.0	27.4	28.2
dat_100_2000	32.0	1.8	0	9.4	3.7	11.3
dat_300_300	0	89.0	78.1	69.4	61.0	43.4
dat_300_1000	0	30.5	22.8	24.7	26.9	27.1
dat_500_1000	0	29.5	25.3	23.7	27.7	28.0
dat_800_0	0	46.7	34.6	24.3	16.5	5.5
dat_800_5000	0	25.0	25.7	29	26.4	28.6

Média de Iterações

A média de diferentes iterações de cada algoritmo. Como o guloso é chamado apenas uma vez, esta será sempre 1. O guloso randomizado possui o máximo de 500 iterações para cada uma das 10 execuções feitas por cada alfa, e a média é calculada com base nestas.

Instâncias	**Guloso**	**Alfa (0.1)**	**Alfa (0.2)**	**Alfa (0.3)**	**Alfa (0.5)**	**Alfa (0.7)**
dat_50_50	1	347.2	320	354.6	310.7	269.5
dat_50_100	1	299	349.8	274	343.1	303.3
dat_50_500	1	285.3	287.9	271.6	271.4	257.8
dat_100_250	1	357.2	356	299.5	319.9	405.7
dat_100_2000	1	242.8	276.5	231.7	286.2	240.1
dat_300_300	1	313	302.7	397.8	305.5	303.9
dat_300_1000	1	320.3	349.9	350.5	280.1	358.7
dat_500_1000	1	316.7	329.1	312.2	315.6	334.5
dat_800_0	1	291.4	317.8	331.9	353.1	321.3
dat_800_5000	1	346.3	283.5	311	331.9	317.8

A partir da análise da tabela de resultados médios, pode-se constatar que em instâncias com mesmo número de vértices, porém de arestas diferentes, o algoritmo encontra melhor resultado naqueles com maior número de arestas. Ademais, o guloso é geralmente encontra resultados melhores do que o randomizado. Em duas instâncias em específico (.dat_50_500 e .dat_100_2000) o randomizado obteve melhores resultados.

Percebe-se também na tabela de desvio percentual da média dos resultados, as instâncias nas quais o algoritmo guloso randomizado obteve melhores soluções do que o guloso (.dat_50_500 e .dat_100_2000) são também as que possuem os menores desvios percentuais no guloso randomizado, isso se deve ao fato de que as médias dos resultados são próximas ao melhor valor.

À medida que cresce o número de vértices das instâncias analisadas, o tempo médio de execução também se eleva exponencialmente, mas há pouca alteração com o aumento do número de arestas.

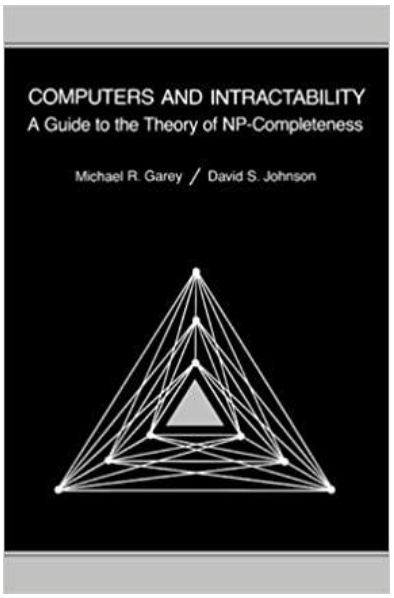
Já na tabela da média de iterações que cada algoritmo executa, nota-se também nas mesmas instâncias que o resultado foi encontrado em um número menor de iterações.

5 Conclusões e trabalhos futuros

O trabalho constituiu no entendimento prático de diversos algoritmos utilizados para problemas com grafos, bem como compreender mais sobre problemas intratáveis (nesse caso, o do Subconjunto Dominante Mínimo), utilizando os algoritmos guloso e guloso randomizado para se registrar e analisar os resultados que estes teriam na execução deste problema.

Para uma nova proposta de abordagem, poderiam ser utilizados novos algoritmos para a ordenação da lista de candidatos (como ao obter um vértice na lista de candidatos remover junto a ele todos os seus adjacentes, por exemplo), desta forma chegando a melhores resultados.

5 Bibliografia e Referências



COMPUTERS AND INTRACTABILITY (Garey & Johnson 1979).

http://www.wpccg.pro.br/apresentacoes/2016/filipe-silva