Universidade Federal de Juiz de Fora

Departamento de Ciência da Computação

Teoria dos Grafos

Algoritmos Construtivos para o Problema do Conjunto 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 2019

# Introdução

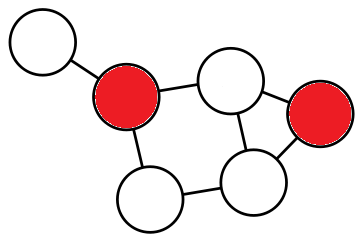
Foi escolhido o problema do Conjunto Dominante Mínimo para um dado grafo;

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.

# 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, com seus membros pintados de vermelho.

Problemas NP-difíceis de grande relevância de Conjunto Dominante Mínimo estão em aplicações relacionadas ao projeto 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.¹

# 3 Abordagens gulosas para o problema

## 3.1 Algoritmo guloso

Primeira coisa a ser feita é ordenar uma lista de candidatos – vértices de maior a menor grau por *Quicksort* feito pelo Marquin;

Inicia na primeira posição da lista de candidatos, removendo-a e adicionando-a a uma lista de solução. Diminui-se um grau de cada vértice adjacente ao removido. Caso o grau de determinado vértice chegue a 0, este é removido da lista de candidatos.

Faz-se então a verificação na lista de graus de se algum conjunto dominante foi encontrado. Se for encontrado, o algoritmo guloso encerra; caso contrário, a lista de candidatos é reordenada e o algoritmo repete o processo.

## 3.2 Algoritmo guloso randomizado

A heurística deste algoritmo é a mesma do Guloso normal, porém há um conjunto de alfas que randomizam a posição a ser escolhida na lista de candidatos. Para cada valor de alfa o algoritmo será chamado 10x e dentro deste há um número máximo de iterações (500). Caso dentro de 200 iterações não se ache uma solução melhor do que a melhor que já foi encontrada, o código retorna a melhor.

# 4 Experimentos computacionais

Nesta seção você deve descrever todo o experimento computacional. Para tanto, defina subseções para:

## 4.1 Descrição das instâncias

As instâncias foram passadas pelo professor orientador, e foram utilizados todas as instâncias.

**Valores Médios**

O valor médio encontrado em cada uma das 10 execuções 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 para 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 |

**Desvio Padrão do Tempo de Execução (em segundos)**

Demonstra o desvio padrão do tempo gasto para a execução de cada algoritmo com base nas instâncias. A segunda coluna exibe o tempo médio para o algoritmo guloso, e as seguintes exibem o tempo médio das 10 execuções de cada valor alfa no randomizado.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instâncias** | **Guloso** | **Alfa (0.1)** | **Alfa (0.2)** | **Alfa (0.3)** | **Alfa (0.5)** | **Alfa (0.7)** |
| dat\_50\_50 | 0.001 | 0.0409 | 0.0374 | 0.0412 | 0.0385 | 0.0304 |
| dat\_50\_100 | 0.001 | 0.0394 | 0.0479 | 0.0368 | 30.0479 | 0.0411 |
| dat\_50\_500 | 0.001 | 0.0564 | 0.0487 | 0.0427 | 0.0419 | 0.0405 |
| dat\_100\_250 | 0.001 | 0.2213 | 0.2256 | 0.1927 | 0.2094 | 0.2711 |
| dat\_100\_2000 | 0.001 | 0.1648 | 0.1703 | 0.1446 | 0.1701 | 0.1413 |
| dat\_300\_300 | 0.009 | 3.1989 | 2.7497 | 3.4152 | 2.5274 | 2.491 |
| dat\_300\_1000 | 0.008 | 2.5413 | 2.8031 | 2.863 | 2.353 | 3.0776 |
| dat\_500\_1000 | 0.026 | 6.8982 | 7.3225 | 7.1136 | 7.4033 | 8.0045 |
| dat\_800\_0 | 0.13 | 47.4115 | 45.0115 | 43.9734 | 45.3874 | 40.3021 |
| dat\_800\_5000 | 0.078 | 34.5436 | 28.7404 | 31.7602 | 37.5538 | 37.8585 |

**Média do Desvio Padrão**

Média calculada entre os desvios padrões de cada uma das 10 execuções de cada algoritmo. Como o resultado do algoritmo guloso é sempre o mesmo, o seu desvio padrão será 0. Para as instâncias randomizadas, o valor está em cada coluna.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instâncias** | **Guloso** | **Alfa (0.1)** | **Alfa (0.2)** | **Alfa (0.3)** | **Alfa (0.5)** | **Alfa (0.7)** |
| dat\_50\_50 | 0 | 0.916515 | 1.16619 | 0.489898 | 1.32665 | 1.36886 |
| dat\_50\_100 | 0 | 1.18743 | 1.13578 | 1.73494 | 1.34536 | 1.53623 |
| dat\_50\_500 | 0 | 0.3 | 0.3 | 0 | 0 | 0.489898 |
| dat\_100\_250 | 0 | 1.35647 | 2.15407 | 1.95959 | 2.45764 | 1.7 |
| dat\_100\_2000 | 0 | 0.489898 | 0.458258 | 0.4 | 0.5 | 0.3 |
| dat\_300\_300 | 0 | 1.93907 | 2.65518 | 2.53969 | 4.32319 | 3.78946 |
| dat\_300\_1000 | 0 | 3.68782 | 5.38888 | 5.59375 | 9.992 | 5.78705 |
| dat\_500\_1000 | 0 | 1.49666 | 1.57797 | 5.86174 | 2.03961 | 2.56125 |
| dat\_800\_0 | 0 | 2.5318 | 3.35261 | 1.84391 | 3.42929 | 3.64555 |
| dat\_800\_5000 | 0 | 13.8798 | 10.6775 | 9.9925 | 9.89747 | 10.2396 |

**Média de Iterações**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 |

## 4.1 Ambiente computacional do experimento e conjunto de parâmetros

Foram utilizadas duas máquinas:

* Uma rodando no Windows 10 – Processador Intel i5 4440, 8 GB RAM;
* Uma rodando no 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 10 iterações.

## 4.2 Resultados quanto à qualidade e tempo

Apresente aqui os resultados quanto à qualidade (valor da função de otimização). Explique o significado das colunas da tabela. Lembre-se de por em negrito os valores associados aos melhores resultados para cada instância. A tabela 4.3 é um exemplo de apresentação dos resultados.

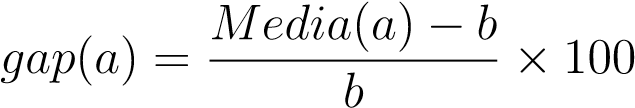
Assim, as heurísticas de construção foram executadas para cada instância em duas diferentes formas: gulosas e gulosas randomizadas reativas. A Tabela 4.3 exibe os resultados gerados pelos algoritmos de construção guloso (colunas 3 e 4), o guloso randomizado reativo (colunas 5 e 6) em comparação com as soluções produzidas pelo algoritmo SCIFI (colunas 7 e 8) para as instâncias. Os valores destacados em negrito indicam os melhores resultados.

Com o objetivo de realizar os testes comparativos, foi utilizado um conjunto de 30 instâncias, sendo as 9 primeiras consideradas pequenas (10 APs e 50 clientes); as instâncias de 10 a 21 consideradas médias (40 APs e 200 clientes); e as 9 últimas, numeradas de 22 a 30, consideradas grandes (70 APs e 350 clientes).

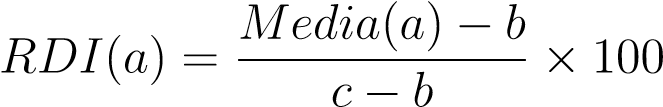
Nos resultados apresentados na Tabela 4.3, a primeira coluna indica o número da instância utilizada, a segunda coluna mostra o valor da melhor média das soluções encontrada dentre todas as execuções realizadas para os diferentes algoritmos apresentados.

As demais colunas da tabela fornecem detalhadamente as métricas resultantes das execuções de cada abordagem, como o desvio relativo da média das soluções do algoritmo com relação à melhor média para a instância (coluna 2), e o tempo médio de execução, dado em segundos.

A princípio, a diferença percentual média (*gap*) das soluções poderia ser calculada conforme a Equação 6 descrita a seguir, onde *Media*(*a*) é o valor médio das soluções obtidas pelo algoritmo *a* e *b* é o valor da melhor média obtida dentre todos os algoritmos comparados (*best*):

 (6)

No entanto, casos em que *b* = 0, ou seja, a melhor solução encontrada possui custo igual a 0, o cálculo do *gap* realiza uma divisão por zero causando um erro matemático. Para resolver esse problema, [2] propõem o uso da métrica denominada Índice de Desvio Relativo (RDI, do inglês *Relative Deviation Index*), cujo cálculo é apresentado pela Equação 7 e define a qualidade da solução de uma dado algoritmo *a*:

 (7)

onde *c* é o valor da pior solução dentre os algoritmos utilizados, e nesse caso específico, a pior média encontrada. Note que *RDI*(*a*) ∈ [0*,*100] e quanto menor o valor resultante, maior é a qualidade da solução do algoritmo *a*.

Segue um exemplo superficial de análise de resultados a partir de uma tabela:

de Analisando a Tabela 4.3 é possível verificar que os resultados quanto à qualidade da solução apresentados pelo algoritmo proposto em sua fase de construção são melhores que aqueles obtidos pelo algoritmo SCIFI. Isto vem a confirmar a hipótese de que o processo de *clusterização* realizado pelo algoritmo através da identificação das componentes conexas do grafo, de fato, impacta na qualidade da alocação de canais com menor interferência.

Nota: Nesta seção você precisa descrever os resultados, não apenas apresentar as tabelas. Procure destacar quais os melhores algoritmos, destacar se há alguma característica de

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instância | Melhor | Guloso | | Reativo | | | SCIFI | |
| *RDI* | Tempo | *RDI* | Tempo | *RDI* | | Tempo |
| st323\_70a | 0,0 | 0,00 | 0,282 | 0,00 | 0,141 | 0,00 | | 0,114 |
| proB789\_100a | 0,0 | 0,00 | 0,129 | 0,00 | 0,140 | 0,00 | | 0,128 |
| lin884\_318 | 0,0 | 0,00 | 0,112 | 0,00 | 0,131 | 100,00 | | 0,128 |
| pcb2208\_442 | 88,3 | 58,89 | 0,115 | 0,00 | 0,089 | 100,00 | | 0,112 |
| pr5314 | 103,0 | 57,45 | 0,111 | 0,00 | 0,092 | 100,00 | | 0,144 |
| wath4180 | 42,7 | 49,23 | 0,111 | 0,00 | 0,087 | 100,00 | | 0,112 |
| lin41817\_710 | 0,0 | 0,00 | 0,098 | 0,00 | 0,115 | 0,00 | | 0,159 |
| kro121002 | 82,0 | 93,48 | 0,114 | 0,00 | 0,115 | 100,00 | | 0,159 |
| dilc | 49,1 | 24,05 | 0,124 | 0,00 | 0,104 | 100,00 | | 0,127 |
| pro789\_100a | 222,4 | 18,26 | 0,150 | 0,00 | 0,202 | 100,00 | | 0,127 |

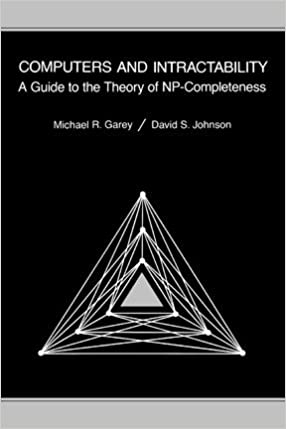
Tabela 2: Exemplo de tabela com comparação de resultados dos algoritmos de construção com o algoritmo da literatura SCIFI

uma ou mais instâncias (tamanho da instância, densidade de arestas ou grupo de instância específico) que esteja influenciando o comportamento de um ou outro algoritmo etc.

# Conclusões e trabalhos futuros

# Referências

  
[http://www.wpccg.pro.br/apresentacoes/2016/filipe-silva.](http://www.wpccg.pro.br/apresentacoes/2016/filipe-silva)

  
COMPUTERS AND INTRACTABILITY (Garey & Johnson 1979).