# Relatórios de Estudo Dirigido (MC032)

**Fernando Morato**

Supervisor: Prof. Pedro J. de Rezende

# Relatório 1

26 de agosto de 2022

**Resumo.** Este relatório consiste das atividades da disciplina MC032 sobre o problema *Graph Burning* (`GB`) realizadas entre os dias 15 e 26 de agosto de 2022. Os principais pontos estão divididos em seções.

## 1.1 Fase de construção do `GRASP`

Realizei uma implementação de um protótipo de fase de construção de soluções viáveis para o `GB`, combinando critérios de gula e aleatorização de acordo com a versão padrão do `GRASP`. A *lista de candidatos* (CL) corresponde ao conjunto dos elementos (vértices) que ainda podem ser adicionados ao final da solução parcial, isto é, ao final da *burning sequence* em construção. Por sua vez, a lista restrita de candidatos (RCL) contém os elementos pertencentes à CL que possuem "alta qualidade" de acordo com um critério estipulado. A cada passo da construção, atualizamos o benefício de cada vértice e construímos uma nova RCL, a partir da qual escolhemos aleatoriamente (com distribuição de probabilidade uniforme) um elemento para fazer parte da solução parcial. [4].

Nessa primeira implementação do `GRASP`, definimos o benefício de um vértice por meio de uma função $g(c)$ que indica a contribuição de cada candidato $c \in \text{CL}$. Inicialmente, optamos por utilizar $g(c) = grau(c)$. Sejam $gmax = \max\{g(c) \mid c \in \text{CL}\}$, $gmin = \min\{g(c) \mid c \in \text{CL}\}$ e $\alpha \in [0, 1]$ um parâmetro real. Um elemento $c$ pertence à RCL se, e somente se, $g(c) \geq gmax - \alpha \cdot (gmax - gmin)$. Portanto, podemos concluir que o valor de $\alpha$ determina o quão gulosa ou aleatória é a escolha, especificamente, quanto menor o valor de $\alpha$ maior é a gula e quanto maior for esse valor, maior é a aleatoriedade da escolha de um elemento na RCL resultante.

Saliento que, ao longo da construção da *burning sequence*, simulamos a propagação do fogo no grafo a partir da solução parcial construída até aquele momento. Essa simulação é feita apenas uma vez a cada construção completa, de maneira semelhante ao procedimento adotado nas heurísticas `GRASP` de [1] para o *Perfect Awareness Problem*. Nesse caso, mantemos em memória um *snapshot* do estado final da propagação ocasionada pela *burning sequence* parcial associada ao $p$-ésimo passo da construção. Se a sequência

é viável, a fase de construção é encerrada e a sequência é devolvida pelo algoritmo. Caso contrário, avançamos ao $(p+1)$-ésimo passo da construção, escolhendo o próximo vértice da sequência e dando continuidade à propagação a partir do referido *snapshot*. Ou seja, o algoritmo da fase de construção devolve *a primeira* solução viável encontrada.

## 1.2 Experimentos computacionais

Para a realização de testes, separei um subconjunto de instâncias mencionadas na literatura que já foram empregadas em experimentos com outras heurísticas. As instâncias escolhidas foram as seguintes:

1. The Network Data Repository
    - Cite-DBLP
    - Mahindas
    - Netscience
    - Polblogs
    - Reed98
2. Stanford large network dataset collection (SNAP Dataset)
    - Gemsec-Deezer (HR)
    - Chameleon-Wikipedia
    - Crocodile-Wikipedia
    - Squirrel-Wikipedia
    - Ego-Facebook
    - Government-Facebook
    - Politician-Facebook
    - TVshow-Facebook

Essa escolha foi feita com o objetivo de compararmos o nosso algoritmo com as heurísticas apresentadas em [3]. Reitero que ainda há outras instâncias que podemos utilizar para efeitos comparativos, mas infelizmente não tive tempo hábil para aprimorar o *script* de leitura de instâncias e fazê-lo ser capaz de ler outros formatos de *input*.

A Tabela 1.1 descreve o tamanho das instâncias. Na Tabela 1.2, comparamos os valores das melhores soluções encontradas pelo `GRASP` e das soluções reportadas pelas heurísticas *Backbone Based Greedy Heuristic* (`BBGH`), *Improved Cutting Corners Heuristic*

(ICCH) e *Component Based Recursive Heuristic* (CBRH). Por fim, comparamos os tempos de execução[1][2] dos algoritmos na Tabela 1.3.[3][4] Em cada execução, foram realizadas 1000 iterações de fase de construção do GRASP com $\alpha = 0.5$.[5] Posteriormente, testaremos outros valores de $\alpha$ e incluiremos outras instâncias no experimento.

| **Origem** | **Nome** | **\|V\|** | **\|E\|** |
|---|---|---|---|
| Network data repository | Cite-DBLP | 12.6K | 49.6K |
| | Mahindas | 1258 | 7513 |
| | Netscience | 379 | 914 |
| | Polblogs | 643 | 2K |
| | Reed98 | 962 | 18K |
| SNAP Dataset | Gemsec-Deezer (HR) | 54K | 498K |
| | Chameleon-Wikipedia | 2.2K | 31.4K |
| | Crocodile-Wikipedia | 11.6K | 170K |
| | Squirrel-Wikipedia | 5K | 198K |
| | Ego-Facebook | 4K | 88K |
| | Government-Facebook | 7K | 89.4K |
| | Politician-Facebook | 5.9K | 41.7K |
| | TVshow-Facebook | 3.8K | 17.2K |

Tabela 1.1: Tamanho das instâncias escolhidas.

| **Origem** | **Nome** | **BBGH** | **ICCH** | **CBRH** | **GRASP** |
|---|---|---|---|---|---|
| Network data repository | Cite-DBLP | **41** | **41** | **41** | 44 |
| | Mahindas | **5** | **5** | **5** | 6 |
| | Netscience | 7 | 7 | 7 | **6** |
| | Polblogs | **6** | **6** | **6** | **6** |
| | Reed98 | **4** | **4** | **4** | **4** |
| SNAP Dataset | Gemsec-Deezer (HR) | **7** | **7** | **7** | **7** |
| | Chameleon-Wikipedia | **6** | **6** | **6** | **6** |
| | Crocodile-Wikipedia | **6** | **6** | **6** | **6** |
| | Squirrel-Wikipedia | **6** | **6** | **6** | **6** |
| | Ego-Facebook | **4** | **4** | **4** | 5 |
| | Government-Facebook | **6** | **6** | **6** | 7 |
| | Politician-Facebook | **7** | **7** | **7** | **7** |
| | TVshow-Facebook | **10** | **10** | **10** | 11 |

Tabela 1.2: Comparação dos tamanhos das *burning sequences* encontradas.

[1] PJR: Qual o valor de se comparar esses tempos? Sabemos quantas iterações foram necessárias para encontrarmos a solução do GRASP?

[2] LFM: Felipe me alertou sobre isso em uma de nossas discussões sobre a comparação de performances mas eu acabei esquecendo de alterar isso. No próximo relatório eu coloco essa informação em uma nova coluna da tabela.

[3] PJR: Recomendo adotar a notação de mm:ss e indicar no caption tratar-se de minutos e segundos. Ficam mais fácil a leitura e as comparações.

[4] LFM: OK!

[5] PJR: Precisamos conversar sobre a elaboração de uma metodologia de desenvolvimento, projeto de experimentos, forma de análise de resultados, etc.

| Origem | Nome | BBGH | ICCH | CBRH | GRASP |
|---|---|---|---|---|---|
| Network data repository | Cite-DBLP | 39s | **22s** | 2m | 2m |
| | Mahindas | **<1s** | 3s | 23s | 6s |
| | Netscience | **<1s** | **<1s** | 1s | 1s |
| | Polblogs | **<1s** | 1s | 2s | 2s |
| | Reed98 | **3s** | **3s** | 5s | 10s |
| SNAP Dataset | Gemsec-Deezer (HR) | **2m 36s** | 7m | 47m | 11m |
| | Chameleon-Wikipedia | 20s | **16s** | **16s** | 23s |
| | Crocodile-Wikipedia | 2m 36s | **42s** | 4m | 2m 34s |
| | Squirrel-Wikipedia | 40s | **34s** | 1m 40s | 2m 26s |
| | Ego-Facebook | 17s | **16s** | 22s | 44s |
| | Government-Facebook | **20s** | 50s | 32s | 1m 19s |
| | Politician-Facebook | **14s** | 32s | 17s | 43s |
| | TVshow-Facebook | **7s** | 22s | 2m | 21s |

Tabela 1.3: Comparação dos tempos de execução.

Como podemos ver, conseguimos soluções próximas das soluções obtidas pelas melhores heurísticas apresentadas na literatura. Contudo, é preciso fazer algumas observações.

Tendo em vista que essa foi a minha primeira implementação de um `GRASP`, a escolha do critério guloso, que é baseada unicamente no grau remanescente dos vértices, foi feita por uma razão prática: é algo fácil de implementar. Contudo, eu suspeito que esse critério não funcione bem para outras instâncias.

Além disso, os tempos de execução do `GRASP` reportados na Tabela 1.3 podem ser aprimorados, visto que há espaço para especializar as estruturas de dados envolvidas no processo de construção com o critério de gula atual. Observe que nesse primeiro momento, eu concentrei meus esforços em implementar uma fase de construção adequada para o *Graph Burning* a fim entender, na prática, o funcionamento da meta-heurística. De fato, não me preocupei muito com o aspecto da eficiência, pois não apostava no sucesso do critério de gula escolhido.[6] [7]

## 1.3 Próximos Passos

Primeiramente, pretendo estender o experimento atual, contemplando outras instâncias usadas na literatura. Em paralelo, planejo investigar critérios gulosos que envolvem medidas centralidade. Inclusive, algumas dessas medidas já foram utilizadas em outras heurísticas. Eis exemplos de medidas de centralidade apresentadas em [6]:

- *Closeness Centrality*

[6] PJR: Essas foram suposições bastante sensatas!
[7] LFM: :)

- *Betweness Centrality*
- *Eigenvector Centrality*
- *PageRank Centrality*

Também pretendo ler os artigos [2] e [5] que apresentam outras heurísticas e instâncias as quais adicionarei ao conjunto de testes.[8] [9]

[8] PJR: Ótimos planos! Paralelizar a implementação com o "estar em dia com a literatura" é uma um excelente abordagem para investigar qualquer problema. Afinal: para fazer o que ninguém fez ainda, é preciso saber o que os outros já fizeram!

[9] LFM: :)

# Referências Bibliográficas


[1] F. de C. Pereira, P. J. de Rezende, and C. C. de Souza. Effective heuristics for the perfect awareness problem. In C. E. Ferreira, O. Lee, and F. K. Miyazawa, editors, *Proceedings of the XI Latin and American Algorithms, Graphs and Optimization Symposium, LAGOS 2021, Online Event / São Paulo, Brazil, May 2021*, volume 195 of *Procedia Computer Science*, pages 489–498. Elsevier, 2021.

[2] Z. R. Farokh, M. Tahmasbi, Z. H. R. A. Tehrani, and Y. Buali. New heuristics for burning graphs. *arXiv preprint arXiv:2003.09314*, 2020.

[3] R. K. Gautam, A. S. Kare, et al. Faster heuristics for graph burning. *Applied Intelligence*, 52(2):1351–1361, 2022.

[4] G. R. Mateus, M. G. Resende, and R. M. Silva. Grasp: Procedimentos de busca gulosos, aleatórios e adaptativos. *2a Escola Luso-Brasileira de Computação Evolutiva*, 2010.

[5] M. Šimon, L. Huraj, I. Dirgová Luptáková, and J. Pospíchal. Heuristics for spreading alarm throughout a network. *Applied Sciences*, 9(16):3269, 2019.

[6] Wikipedia contributors. Centrality — Wikipedia, the free encyclopedia, 2022. [Online; accessed 25-August-2022].