Comparação entre algoritmo Guloso e resolução com CPLEX para uma adaptação do Problema do Bombeiro

Giovany da S. Santos

[1]Instituto de Matemática e Computação – Universidade Federal de Itajubá (UNIFEI)
Itajubá – MG – Brazil

giovany.silva@unifei.edu.br

Abstract. *The Firefighter's Problem is a graph problem that explores the concept of vertex neighborhoods through fire spreading to the neighboring vertices of an already burnt vertex. The Firefighter's Problem has practical applications in fire control, network security, diseases control, among others. In this article, we will see an adaptation of the original problem of the Firefighter's Problem, as well as a greedy implementation and a mathematical formulation for the problem.We will also see a comparison between the results obtained in the various tests for these implementations.*

Resumo. *O Problema do bombeiro é um problema de grafos que explora o conceito de vizinhança de vértices através do espalhamento de incêndio para os vértices vizinhos de um vértice já queimado.O Problema do Bombeiro tem aplicações práticas no controle de incêndios, segurança em redes, controle de doenças entre outros.Nesse artigo veremos uma adaptação do problema original do Problema do bombeiro, além de uma implementação Gulosa e uma formulação matemática para o problema.Será realizada também, uma comparação entre resultados obtidos nos diversos testes para essas implementações.*

1. Introdução

O Problema do Brigadista é um problema no qual há um incêndio com foco inicial em uma região e a cada instante de tempo se propaga para regiões vizinhas.Outra consideração é que quando uma região é queimada ou defendida ela passa a ser intocada, ou seja brigadistas não serão alocados para essa região novamente.O objetivo do problema é minimizar o número de regiões incendiadas até que todo o incêndio seja controlado.

A versão adaptada apresentada nesse artigo coloca que a partir de um número limitado de bombeiros devem ser realizadas diferentes alocações desses funcionários obedecendo às seguintes regras: é fixado um número máximo de vértices defensáveis por rodada e um número de bombeiros que podem ser alocadas para as rodadas.Também nunca haverá uma ocasião de que um bombeiro trabalhe mais que k dias consecutivos por questões de direitos à folga no (k+1)-ésimo dia desse funcionário.

Simplifiquemos as definições vistas em [Ramos 2018] e [Fomin et al. 2016].Na linguagem de Teoria dos Grafos, temos um grafo **G=(V,E),V= {1,...,n}** e um conjunto de arestas **A={(u,v) :u e v** $\in V$,u$\neq$v **}**.A quantidade de vértices pode ser denotada por **n=|V(G)|** e a quantidade de arestas **m = |E(G)|**.Os vizinhos de um vértice v são denotados por **NG(v)**.Um conjunto **B** define o conjunto de vértices queimados e um conjunto **D**

define o conjunto dos vértices defendidos.O incêndio se espalha inicia em um vértice **u** $\in$**V** e se espalha para os vértices **v'** $\in$ **NG(v)**$\setminus$**D**.O objetivo do problema á maximizar |**V(G)**|-|**B**| ou maximizar |**D**|.

2. Revisão Bibliográfica

O problema original do Problema do Bombeiro foi proposto por [Hartnell 1995], [Hartnell and Li 2000] comparou os efeitos de algoritmos gulosos com algoritmos ótimos em árvores para o problema do bombeiro.Provou-se que a solução dos gulosos contemplam pelo menos metade da eficácia da solução de algoritmos ótimos.

[Elliot et al. 2010] propôs a resolução de duas versões do problema do bombeiro: o conceito de espalhamento de doenças e o uso de vacinações.Na primeira versão as doenças eram espalhadas e as vacinações devem proteger os nós ainda não contaminados.Nessa o objetivo é minimizar o número de nós infectados.Na segunda, a vacina foi vista como um meio de infectar os indivíduos.Por essa versão temos um conjunto de nós que devem ser salvos e a meta é minimizar o orçamento utilizado para que a ação seja controlada.Foram propostos algoritmos aproximados para resolver o problema.

[Michalak 2018] propôs um método de busca local para melhorar soluções produzidas por dois algoritmos evolutivos(MOEA / D e o NSGA-II).O algoritmo consistia em partir de uma solução inicial e fazer uma busca local utilizando duas heurísticas : defender o vizinho mais próximo de vértices queimados e por último defender os vizinhos com maior grau de vértices não queimados.No artigo foi mostrado que a combinação de algoritmos evolutivos e heurísticas que podem ser úteis para a resolução do problema. Através de análises de resultados de algumas instâncias com tamanho entre 50 e 200 foi visto que as heurísticas propostas apontavam para soluções boas em comparação com algoritmos de busca local aleatória.

[Sales et al. 2020] propôs uma variante do problema do bombeiro com a introdução de um conceito de resistência que se distingue pela presença de uma resistência ao fogo, inteira associada a cada vértice.Nesta variante, cada vértice v possui inicialmente uma resistência λv.A cada iteração, a resistência de um vértice intocado v adjacente a algum vértice queimado é decrementada do número de vértices queimados adjacentes a ele.No FFPR(*The Firefighter Problem with Resistant Vertices* - O Problema do Brigadista com Vértices Resistentes) um vértice intocado **v** se torna queimado quando λ**v** se torna menor ou igual a 0.Foi estudado heurísticas **k-DynDegree** que consiste em: a cada iteração t, ordenar os vértices de **Nk(Qt)** (vizinhança intocada de Q no tempo t) sendo k um parâmetro a ser escolhido pelo usuário.Primeiro, em ordem não crescente de grau dinâmico e depois em ordem não crescente de distância para os vértices queimados e então defender os **D** primeiros vértices nessa ordem.Dessa maneira serão defendidos os vértices com maior grau dinâmico que estão mais distantes dos vértices queimados.Definiram também uma nova heurística **k-min**λ, tal heurística consiste em a cada iteração t ordenar os vértices **Nk(Qt)** primariamente em ordem crescente de resistência e secundariamente em ordem não crescente de distância para os vértices queimados e então defender os d primeiros vértices nessa ordem.E foi identificado uma última heurística a **k-random** que consiste em selecionar aleatoriamente **D** vértices do conjunto **Nk(Qt)** para serem defendidos.

3. Métodos

É proposto neste artigo, uma formulação matemática adaptada da formulação encontrada em [Ramos 2018].Também foi proposto um algoritmo Guloso para a resolução do problema,uma definição de algoritmo Guloso é feita por[Rocha and Dorini 2004] no qual é visto como um algoritmo que toma decisões nas informações que esse possui sem a preocupação de efeitos que podem vir por causa das suas decisões, e também que esse tipo de algoritmo quando tem o seu correto funcionamento apresenta uma eficiência.Então, foi implementado um algoritmo Guloso que usa as heurísticas de defender os vértices dando prioridade aos que possuem maior grau dinâmico e que estão mais próximos de vértices já queimados.

Após isso, comparou-se os resultados em questões de tempo e valor solução obtido do algoritmo Guloso com a implementação baseada na formulação matemática que, foi implementada no resolvedor CPLEX.

4. Formulação matemática

4.1. Formulação

Na formulação matemática na Figura 1 : **t** representa o instante do incêndio avaliado, **T** é o número de rodadas, que na implementação CPLEX foi setado com o valor do número de vértices, ou seja simulo-se para o limitante de rodadas que, um vértice seria defendido por rodada. **b(v,t),** $\forall$ **v** $\in$ **V** e **0**$\leq$ **t** $\leq$ **T** é a variável para indicar se o vértice **v** está queimado na rodada **t**, enquanto que **d(v,t),** $\forall$**v** $\in$ **V** e **0**$\leq$ **t** $\leq$**T**.**V** define se o vértice está defendido no tempo **t**. **D** representa o número máximo de vértices que podem ser defendidos por rodada e **B** o conjunto de vértices queimados.

$$\max \quad |V| - \sum_{v \in V} b_{v,T} \tag{1}$$

s.a.

$$b_{v,t} + d_{v,t} - b_{v',t-1} \geq 0 \qquad \forall v \in V, v' \in N(v) \text{ e } 1 \leq t \leq T \tag{2}$$

$$b_{v,t} + d_{v,t} \leq 1 \qquad \forall v \in V \text{ e } 1 \leq t \leq T \tag{3}$$

$$b_{v,t} - b_{v,t-1} \geq 0 \qquad \forall v \in V \text{ e } 1 \leq t \leq T \tag{4}$$

$$d_{v,t} - d_{v,t-1} \geq 0 \qquad \forall v \in V \text{ e } 1 \leq t \leq T \tag{5}$$

$$\sum_{v \in V} (d_{v,t} - d_{v,t-1}) \leq D \qquad \text{para } 1 \leq t \leq T \tag{6}$$

$$b_{v,0} = 1 \qquad \forall v \in B \tag{7}$$

$$b_{v,0} = 0 \qquad \forall v \in V \setminus B \tag{8}$$

$$d_{v,0} = 0 \qquad \forall v \in V \tag{9}$$

$$b_{v,t}, d_{v,t} \in \{0,1\} \qquad \forall v \in V \text{ e } 1 \leq t \leq T \tag{10}$$

Figura 1.Formulação matemática

Criamos uma variável **f** (do inglês firefighter), que identifica um bombeiro específico.Onde **f(i,t)** $\in$ $\{0,1\}$ **1**$\leq$ **i** $\leq$ **D**. Adicionamos a restrição de que dado um conjunto de bombeiros, uma quantidade **D** será selecionada desses, para que trabalhem na

rodada.Então, houve uma modificação na restrição (6) para :

$$\sum_{u \in V} (d(v,t) - dv, t-1)$$

$\leq$

$$\sum_{i=1}^{D} f(i,t)$$

,onde indica que os vértices defendidos podem ser no máximo D bombeiros quaisquer.Outra restrição foi adicionada (11) com a notação:

$$\sum_{t'=t}^{t+k-1} f(i,t') \leq K \mid \mathbf{1} \leq t \leq T - K$$

,que limita o bombeiro a trabalhar no máximo K dias consecutivos.

4.2. Resolução de instância exemplo

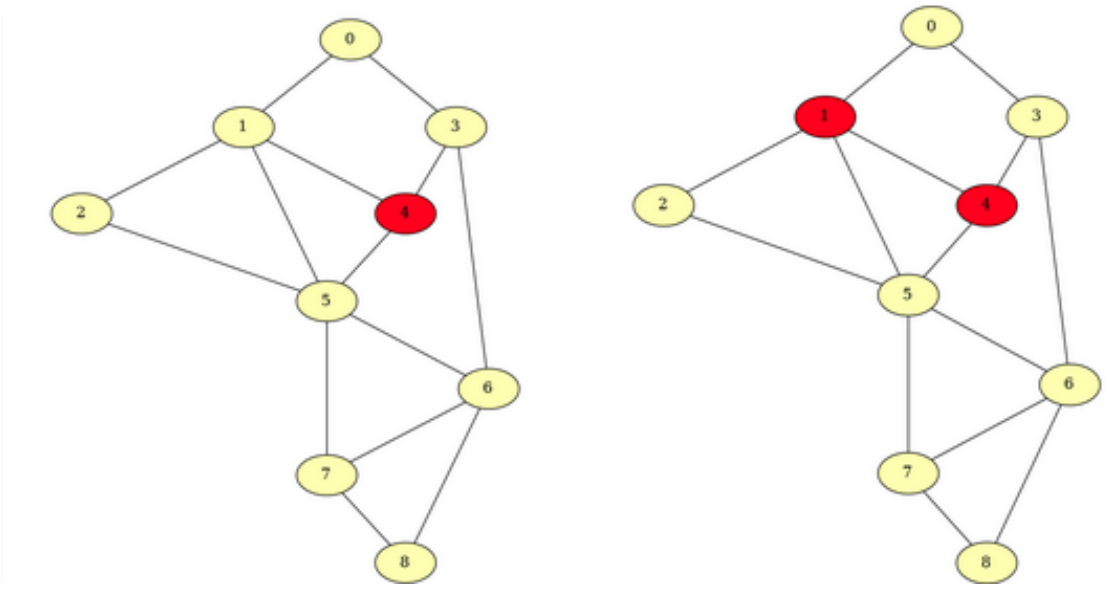

Figura 2.Algoritmo CPLEX - estágio 1 e 2

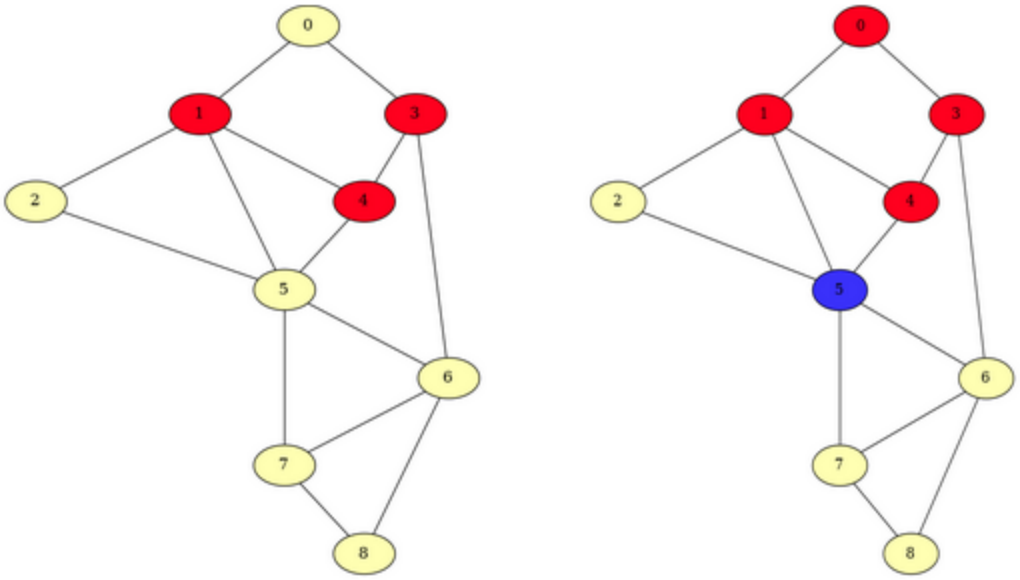

Figura 3.Algoritmo CPLEX - estágio 3 e 4

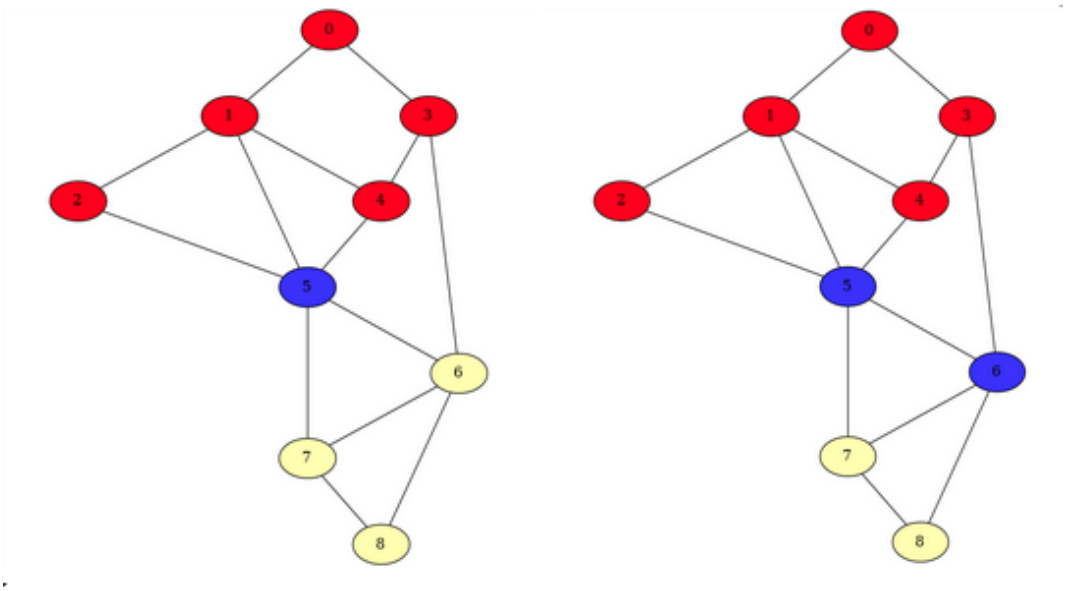

Figura 4.Algoritmo CPLEX - estágio 5 e 6

É possível observar que foi encontrada uma estratégia do bombeiro defender vértices que impediram a passagem das queimadas para outra região.Essa barreira pode ser vista na defesa dos vértices **5** e **6**, sendo que após defendê-los não foi necessário defender os vértices **7** e **8**, já que era certo de que esses vértices nunca mais seriam queimados.Com isso o total de vértices defendidos pode ser contabilizado por |**V(G)**|-|**B**|,ou seja **9-5=4** vértices defendidos.

5. Algoritmo Guloso

5.1. Algoritmo

```
ENTRADA QTD_BOMBEIROS,QTD_VERTICES,MAX_DEFENSAVEL_RODADA,FOCO_INICIAL,MAX_DIAS_CONSEC,GRAFO
SAIDA VALOR SOLUÇÃO E ARRAY SOLUÇÃO

ALGORITMO_GULOSO FIREFIGHTER
    EQUANTO EXISTE_VERTICE FAÇA

      PARA CADA BOMBEIRO FAÇA
        SE BOMBEIRO ATINGIU MAX_DIAS_CONSEC FAÇA
            ZERE A QUANTIDADE DE DIAS TRABALHADOS
        SENAO
            SELECIONE BOMBEIRO E AUMENTE A QUANTIDADE DE DIAS TRABALHADOS EM UM.
        FIMSE

      FIMPARA

      ARMAZENE EM UM ARRAY O GRAU INTOCADO E PROXIMIDADE A UM FOCO DE INCÊNDIO DE CADA VERTICE QUE ESTÁ INTOCADO

      PARA CADA BOMBEIRO FAÇA
          DEFENDA O VERTICE PRIORIZANDO GRAU E DEPOIS A PROXIMIDADE
          ATUALIZE O GRAU INTOCADO DE CADA VÉRTICE ADJACENTE AO VÉRTICE DEFENDIDO
      FIMPARA

      PARA CADA VERTICE FAÇA
          SE VÉRTICE ESTÁ INTOCADO ENTAO
              EXISTE_VERTICE= VERDADE
          FIMSE
      FIMPARA

      SE NAO EXISTE_VERTICE FIMALGORITMO
```

```
30
31          PARA CADA UM DOS VÉRTICES QUEIMADOS FAÇA
32            QUEIME OS SEUS ADJCANTES
33            ATUALIZA_GRAU INTOCADO DE CADA VIZINHO
34            ATUALIZA_PROXIMIDADE DE TODOS OS VERTICES
35          FIMPARA
36  FIMALGORITMO
37
38  ATUALIZA_PROXIMIDADE(VERTICE_QUEIMADO)
39
40    PARA CADA VERTICE V
41      SE V ESTÁ INTOCADO
42        SE DISTANCIA DE V A ESSE VÉRTICE É MENOR QUE A ATUAL ARMAZENADA FAÇA
43            ATUALIZE DISTANCIA
44        FIMSE
45      FIMSE
46    FIMPARA
47  FIMATUALIZA_PROXIMIDADE
48
49  ATUALIZA_GRAU (VERTICE_QUEIMADO)
50    PARA CADA VERTICE V ADJACENTE A VERTICE_QUEIMADO FAÇA
51      DIMINUA O GRAU INTOCADO EM UM.
52    FIM PARA
53  FIMATUALIZA_GRAU
```

Figura 5.Algoritmo Guloso

Na subseção seguinte veremos os passos desse algoritmo guloso para resolver uma instância com nove vértices, foco inicial de incêndio no vértice **4** e com um bombeiro trabalhando

5.2. Resolução de instância exemplo

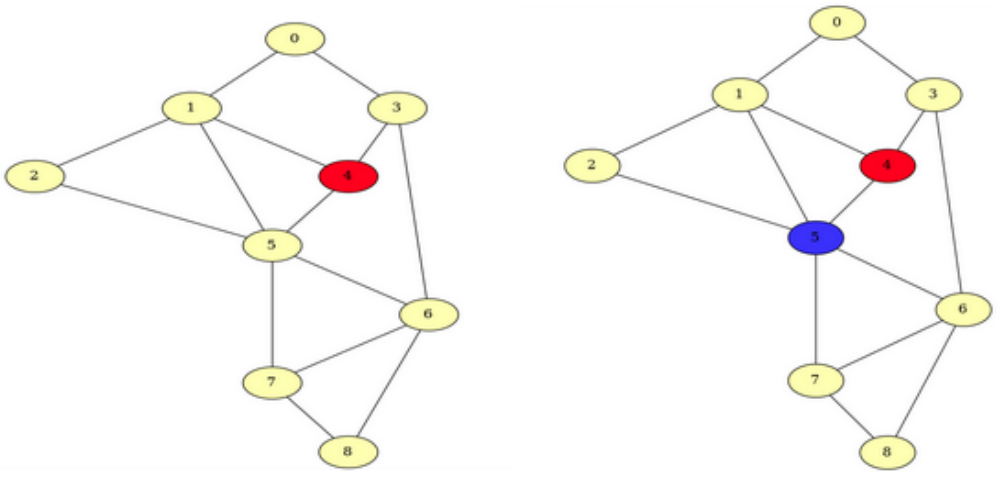

Figura 7.Algoritmo Guloso - estágio 1 e 2

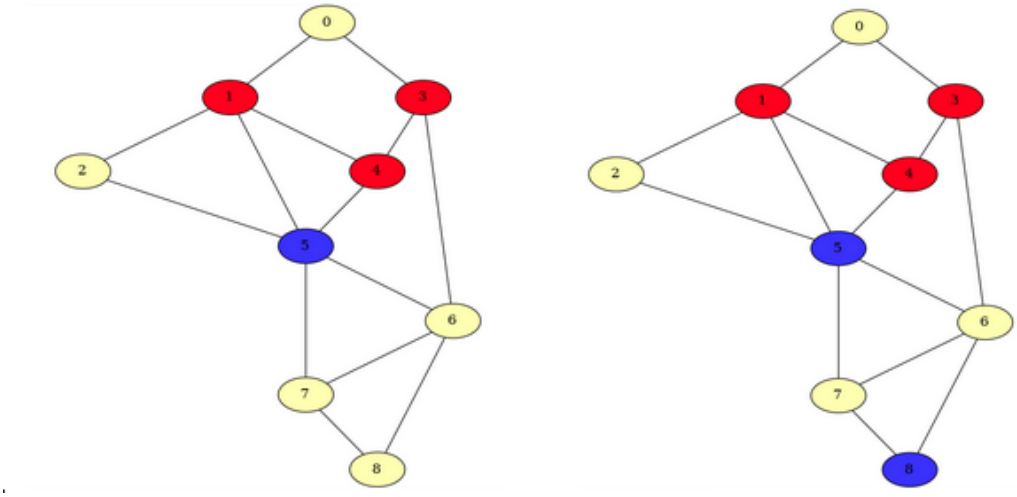

Figura 8.Algoritmo Guloso - estágio 3 e 4

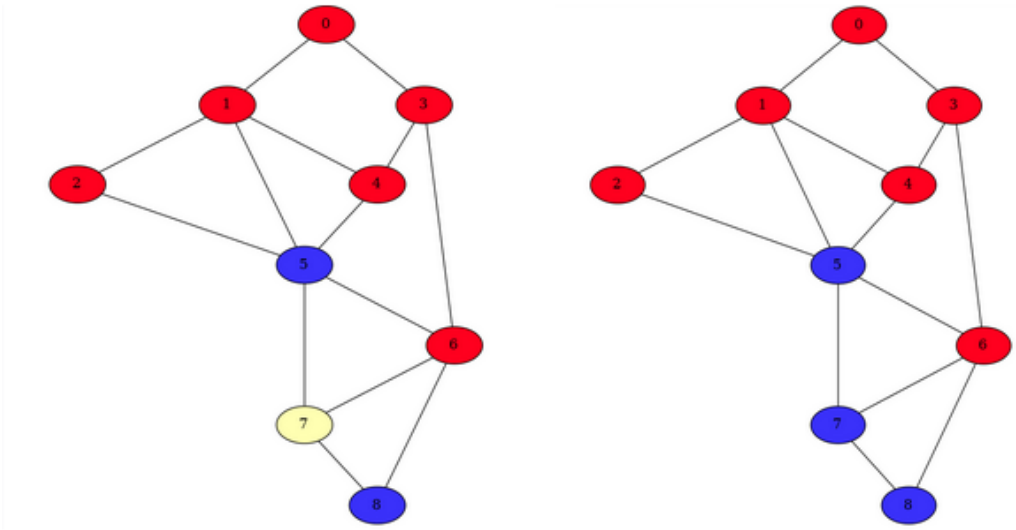

Figura 9.Algoritmo Guloso - estágio 5 e 6

Após o vértice inicial queimado **(4)**, o vértice **5** é defendido pois possui o maior grau intocado, igual a **4**, e está próximo de um foco de incêndio com a menor distância possível, igual a **1**.Após essa rodada, os vizinhos do vértice **4(1 e 3)** também são queimados, e o vértice a ser defendido passa ser o **8**, que pelas regras de prioridade empata com o vértice **7**, mas por motivos de aleatoriedade o algoritmo acaba por selecionar o vértice **8**.Na próxima rodada, os vizinhos do vértice **1 (2 e 0)** são queimados e os vizinhos do **3** também (nesse caso, o **6** que ainda não foi queimado).Por fim, o vértice **7** é defendido e são totalizados **3** vértices defendidos por essa implementação Gulosa.

6. Testes e resultados

Os testes foram realizados em uma máquina com as seguintes configurações

- Processador: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
- Memória RAM: 4 GB
- Sistema operacional: Kali Linux 2020.3
- Kernel do Linux: 5.7.0-kali1-amd64

- Placa de vídeo: Intel Corporation HD Graphics 620 (rev 02)

Para os primeiros testes foram fixadas as variáveis com os seguintes valores:

- Número de bombeiros : 10
- Foco inicial : vértice 1
- Quantidade máxima de vértices defendidos por rodada: 15
- Quantidade máxima de dias consecutivos: 5

No primeiro teste avaliamos o algoritmo guloso para diferentes tamanhos com as seguintes densidades: **10%,50% e 100%**(aproximadamente **100%**), onde uma densidade próxima de zero representa pouco denso e próxima de **100 %** muito denso. Com isso, temos um gráfico a partir dos resultados obtidos, e que mostra o tempo gasto pelo algoritmo guloso para cada uma das instâncias:

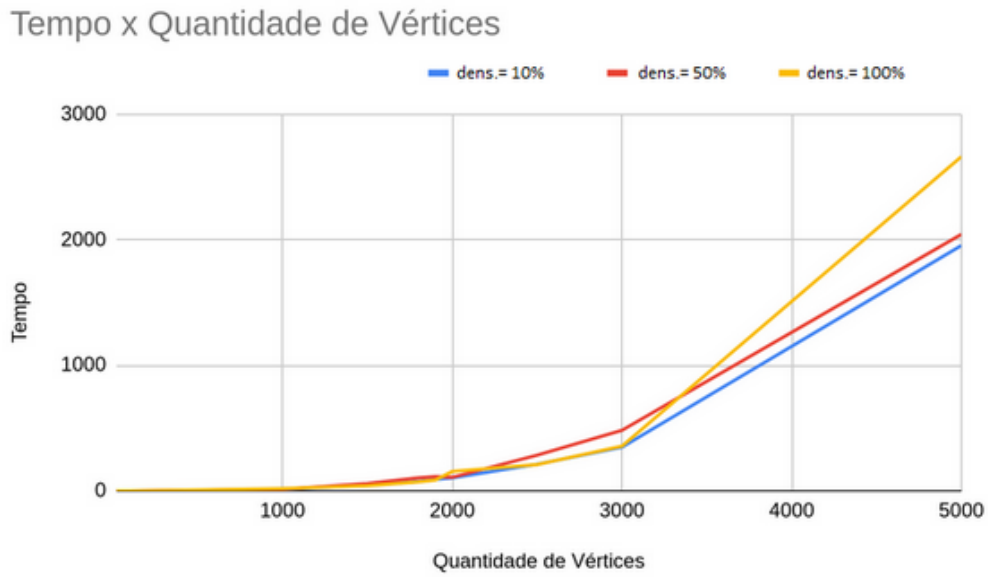

Gráfico 1.Tempo x Quantidade de Vértices

Podemos observar no Gráfico 1 que, as instâncias testadas que possuem um número de vértices até aproximadamente **3000** obtiveram um tempo aproximado para as três densidades avaliadas, porém a instância de **5000** mostrou que com as maiores densidades o tempo gasto do algoritmo também é maior. Também temos o gráfico que contém as soluções encontradas com essas variações de quantidade de vértices e densidade do grafo:

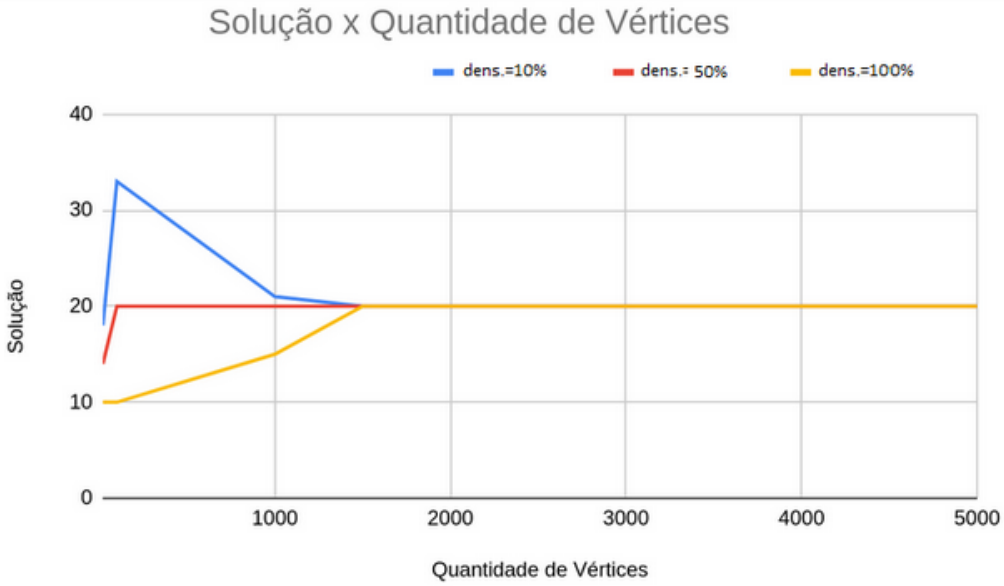

Gráfico 2.Solução x Quantidade de Vértices

Vemos que no Gráfico 2 que, as instâncias menos densas com até **1500** vértices obtiveram uma solução melhor que as mais densas.Além disso, é possível observar que todas as instâncias maiores que **1500** obtiveram a solução encontrada de valor igual a **20**.

Para efeito de comparação de eficiência do guloso com a implementação da formulação matemática com o CPLEX utilizamos instâncias de tamanho de **10** a **100** e variamos as suas densidades de **10%** a **100%**.Visto que, a resolução com o CPLEX não suportou uma quantidade maior de vértices.Uma outra observação é que, nos outros testes foi mantido todas as variáveis anteriormente fixadas, exceto a variável correspondente ao número de bombeiros.

No Gráfico 3, temos o gráfico de soluções ótimas encontradas pelo CPLEX, é possível observar que o valor solução é maior para as instâncias com menores densidades

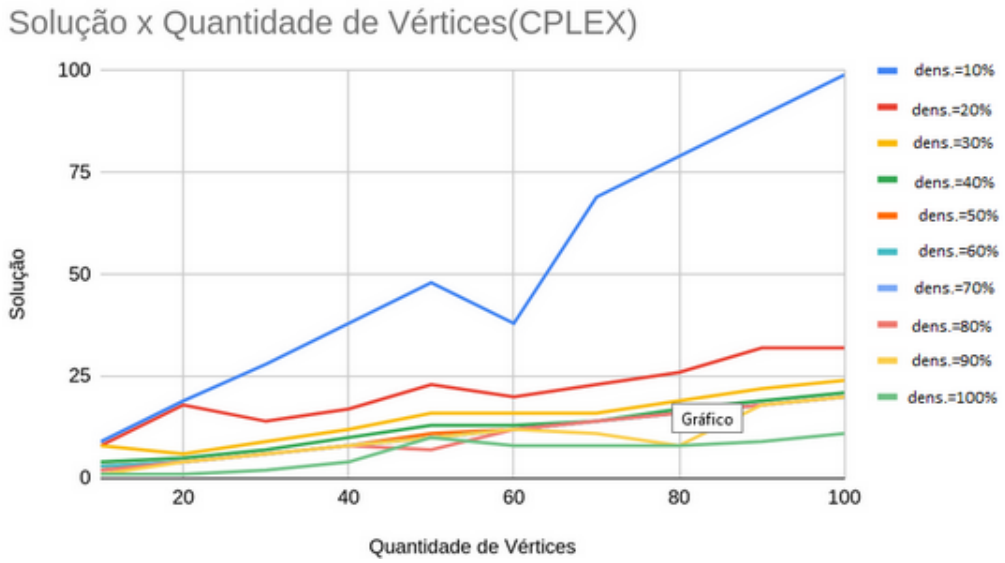

Gráfico 3.Solução X Quantidade de Vértices(CPLEX)

A respeito do tempo gasto, as instâncias testadas que obtiveram variações signi-

ficativas no tempo de execução do algoritmo estão entre **70** e **100** como quantidade de vértices como pode ser visto no Gráfico 4.Outra observação é que os maiores tempos encontrados estão nas instâncias com as seguintes densidades: **50%,60%** e **70%**.

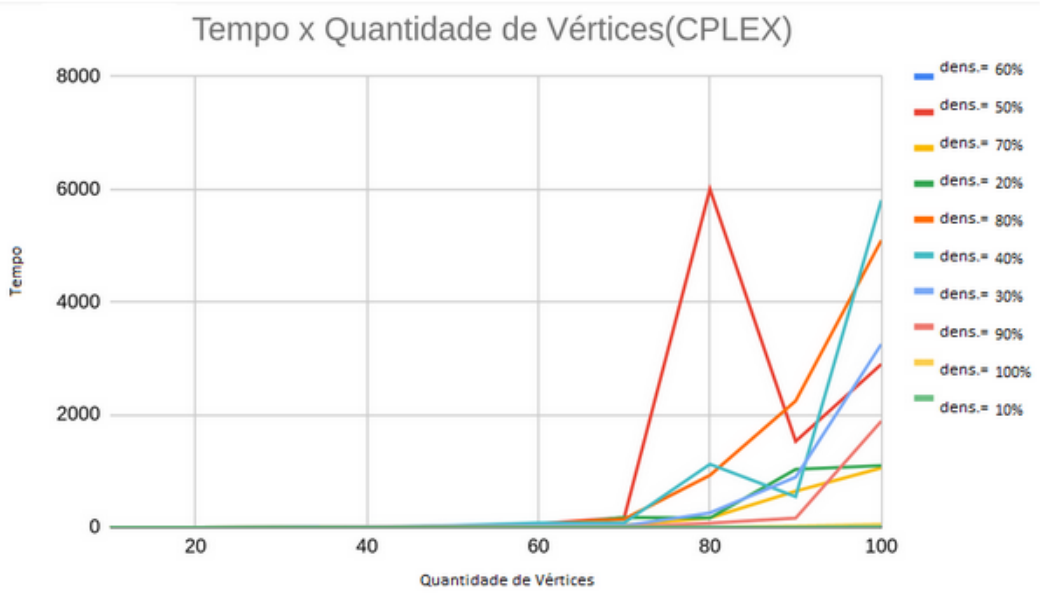

Gráfico 4. Tempo X Quantidade de Vértices(CPLEX)

No Gráfico 5, podemos notar que as melhores soluções estão nas instâncias com menores densidades,igualmente ao ocorrido na implementação CPLEX.Uma das explicações é que instâncias com maiores densidades indicam um número grande de arestas e por consequência, um número maior de vértices com alto grau, e por isso o espalhamento de queimadas ocorrerá em mais vértices.

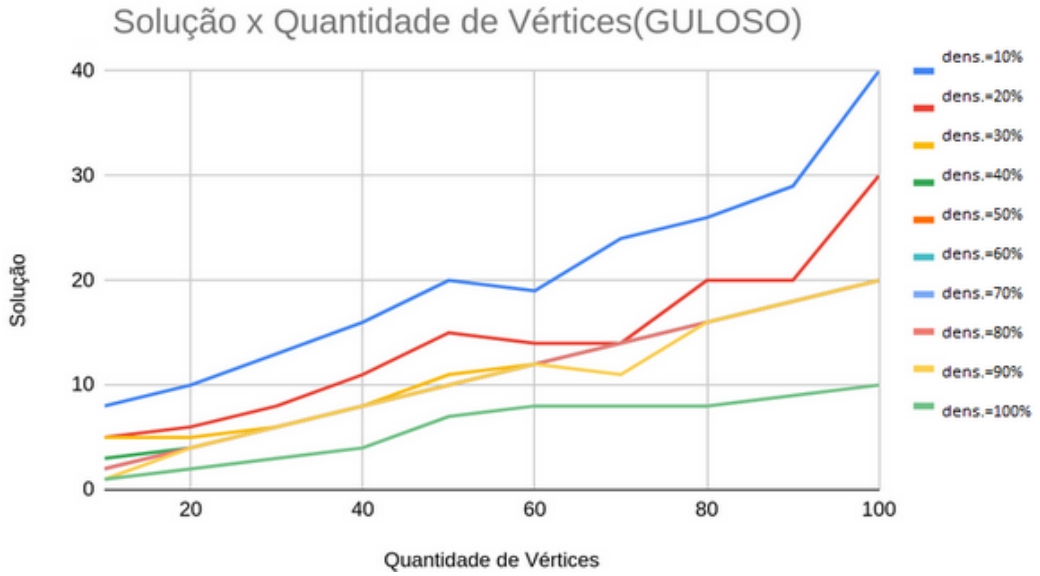

Gráfico 5. Solução X Quantidade de Vértices(GULOSO)

Para fazermos uma melhor análise dos dois algoritmos verificamos a solução média de cada um dos diferentes tamanhos(quantidade de vértices), e plotamos no gráfico os valores médios à medida que a densidade das instâncias foram aumentadas.

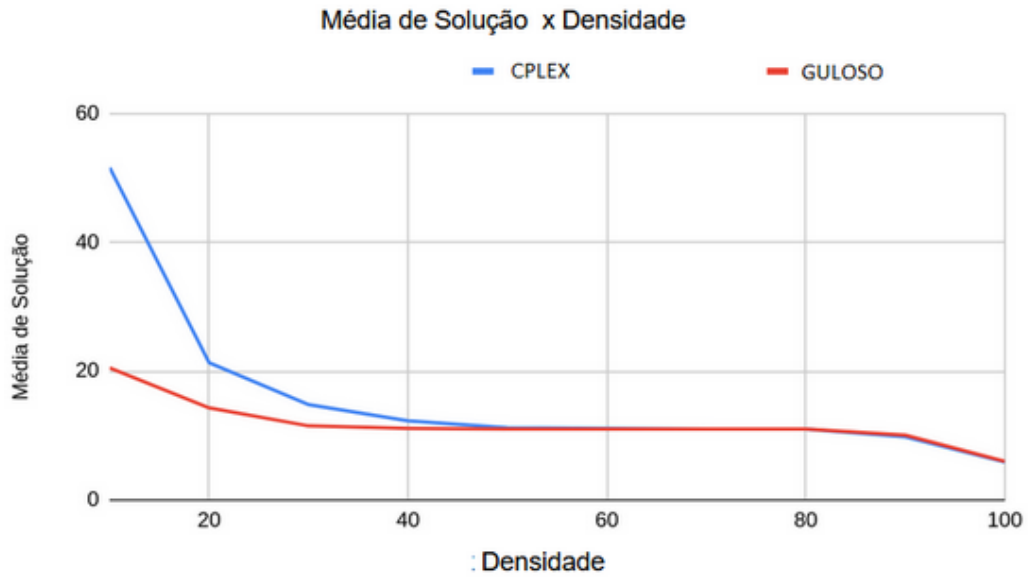

Gráfico 6. Média de Solução X Densidade

Podemos observar no Gráfico 6 que, a solução média teve uma maior diferença, entre o Guloso e CPLEX, nas instâncias com densidades menores que **50%**.Nas densidades maiores que **50%** os valores solução foram iguais ou próximos.

No Gráfico 7 nota-se que, a média de solução para a solução para o CPLEX, considerando a quantidade de vértices, foi maior, porém esteve próxima ao Guloso para todas as diferentes quantidades de vértices.

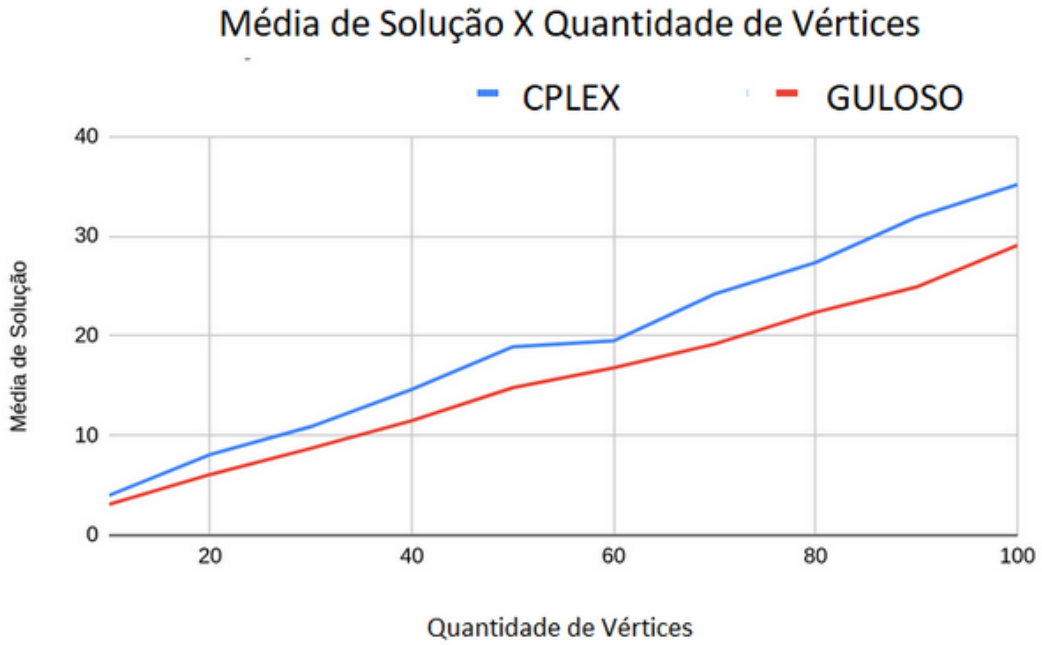

Gráfico 7. Média de Solução X Quantidade de Vértices

O tempo no algoritmo Guloso para instâncias todas as instâncias menores que **100** foi muito próximo de zero, então a linha do Guloso ficaria quase que paralelo ao eixo **X**.Por isso não plotamos no Gráfico 8, onde só está representado o tempo gasto pela implementação CPLEX.

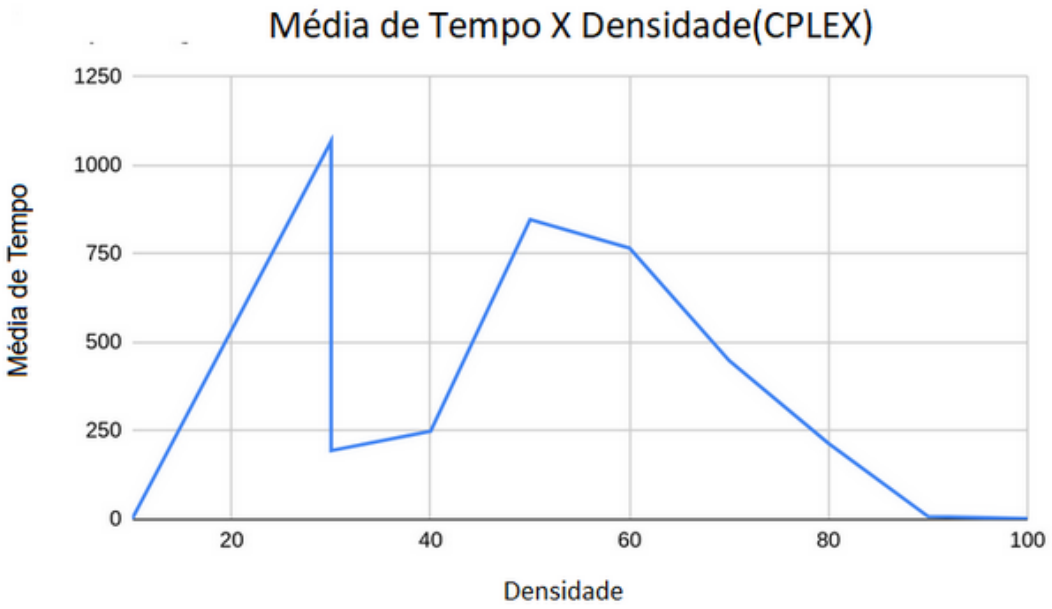

Gráfico 8. Média de Tempo X Densidade(CPLEX)

No Gráfico 9 podemos ver que, a média de tempo quando aumentamos a quantidade de vértices só apresentou uma média de tempo significativa para instâncias maiores que **60**.Além disso, a média de tempo foi maior para as instâncias de tamanho **100**.

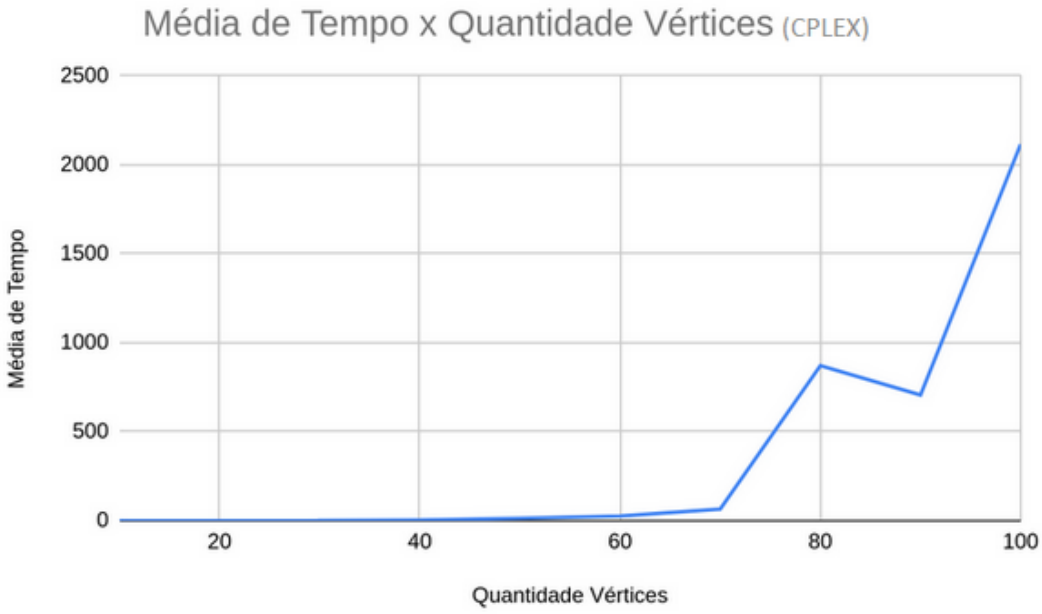

Gráfico 9. Média de Tempo X Quantidade de Vértices(CPLEX)

7. Conclusão

Portanto, para uma quantidade de vértices muito pequenas é muito inviável utilizar a implementação CPLEX, visto que o método guloso apresentou médias de soluções muito próximas a esse e com muito menos tempo.Se faz necessário fortalecer a formulação matemática para avaliar as instâncias com maiores quantidades de vértices, uma melhoria

a ser feita seria no limitante do número de rodadas.Além disso, uma outra observação é que o Guloso mostrou uma piora significativa para instâncias muito grandes, talvez seja possível que uma implementação CPLEX seja viável nesse caso, mesmo que apenas para algumas densidades.

Referências

Elliot, A., Deeparnab, C., Ameya, H., and Chaitanya, S. (2010). Approximability of the firefighter problem. Disponível em: https://www.cs.rpi.edu/ eanshel/papers/firefighter.pdf. Acesso em: 07/11/2020.

Fomin, F. V., Heggernes, P., and Leeuwen, E. J. (2016). The firefighter problem on graph classes. *Theoretical Computer Science*, 613:38–50.

Hartnell, B. (1995). Firefighter! an application of domination. *25th Manitoba Conference on Combinatorial Mathematics and Computing*.

Hartnell, B. and Li, Q. (2000). Firefighting on trees: How bad is the greedy algorithm? *Congressus Numerantium*, pages 187–192.

Michalak, K. (2018). Ed-ls: a heuristic local search for the firefighter problem. *Conference: the Genetic and Evolutionary Computation Conference Companion*.

Ramos, N. (2018). Um estudo computacional do problema do brigadista em grafos. *Universidade Estadual de Campinas*.

Rocha, A. and Dorini, L. B. (2004). Algoritmos gulosos: definições e aplicações. Disponível em: https://www.ic.unicamp.br/ rocha/msc/complex/algoritmosGulososFinal.pdf. Acesso em: 02/12/2020.

Sales, L. F. d. L., Azevedo Neto, R., and Cintra, G. F. (2020). O problema do brigadista com vértices resistentes. Disponível em: https://doi.org/10.5753/etc.2020.11090. Acesso em: 01/12/2020.