

META-HEURÍSTICAS APLICADAS AO PROBLEMA DA REMOÇÃO DE GARRAS EM GRAFOS

Julia Davet de Oliveira¹, Rafael Cavalcante Chaves¹, Thiago Gouveia da Silva¹

¹Instituto Federal da Paraíba, João Pessoa — PB
João Pessoa — PB

Autores para correspondência: julia.davet.oliveira@gmail.com
rafaelchavesaec@gmail.com

RESUMO

Garra é a denominação dada a um grafo bipartido completo $K_{1,3}$, ou seja, que possui uma partição com um vértice e outra com três vértices. Um grafo $G = (V, E)$ é tido como livre de garras se não existirem subgrafos induzidos por vértices em G isomorfos ao grafo $K_{1,3}$. O Problema de Eliminação de Garras (PEG) consiste em encontrar um conjunto de cardinalidade mínima $S \subseteq V(G)$ de tal maneira que $G[V(G) \setminus S]$ seja um grafo livre de garras. O PEG é NP-Difícil, derivando-se a prova genérica de Lewis e Yannakakis (1980) para problemas de eliminação de vértices. Desta maneira, foi proposto dois métodos baseados nas meta-heurísticas Algoritmos Genéticos e VNS, respectivamente. Foram utilizados 27 grupos de instâncias de grafos intervalares e quatro grupos de instâncias de grafos randômicos nos testes computacionais os quais demonstraram que ambos os métodos são capazes de chegar a soluções de alta qualidade com pequeno esforço computacional.

PALAVRAS CHAVE. Grafos Livres de Garras, Algoritmo Genético, VNS.

Área Principal: MH – Metaheurísticas

ABSTRACT

Claw is the denomination of a complete bipartite graph $K_{1,3}$, i.e., it has a partition with one vertex and another with three vertices. A graph $G = (V, E)$ is considered to be free of claws if there are no vertex-induced subgraphs in G isomorphic to the graph $K_{1,3}$. The Graph Declawing Problem (GDP) aims to find a minimum subset $S \subseteq V(G)$ in the way that $G[V(G) \setminus S]$ is a claw-free graph. GDP is NP-Difficult, deriving the generic proof from Lewis e Yannakakis (1980) for vertex elimination problems. In this work, we propose two methods based on the meta-heuristics Genetic Algorithms and VNS, respectively. 27 groups of interval graph instances and 4 groups of random graph instances were used in computational tests, which have shown that both methods are capable of reaching high quality solutions with little computational effort.

KEYWORDS. Claw-Free Graphs, Genetic Algorithm, VNS.

Main Area: MH - Metahuristics

1. Introdução

Garra é a denominação dada a um grafo bipartido completo $K_{1,3}$, que possui uma partição unitária e outra com três vértices. Na literatura é possível encontrar diversos estudos sobre as propriedades dos grafos livres de garras. Podemos citar Minty (1980), com a busca de Conjuntos Independentes com Peso Máximo; Wen-Lian e Nemhauser (1982), com a Cobertura por Cliques de Peso Mínimo em grafos perfeitos livres de garras; além do Problema da Geração de Grafos de Intervalo Unitário com eliminação mínima de vértices (Bevern et al., 2010), onde um grafo não orientado é formado a partir de um conjunto de intervalos em uma reta real, com um vértice para cada intervalo e uma aresta entre os vértices cujos intervalos se cruzam.

Formalmente, tomemos um grafo simples e não orientado $G = (V, E)$, sendo V o conjunto de vértices e E o conjunto de arestas. G é livre de garras quando não contém nenhum subgrafo induzido por vértices isomorfo a um grafo bipartido completo $K_{1,3}$, também chamado de garra. Este grafo G faz parte, então, do grupo de grafos que satisfazem uma propriedade Π específica, neste caso, ser um grafo livre de garras. Quando um grafo não satisfaz Π , surge o problema da modificação deste grafo para que esta propriedade seja satisfeita, neste caso, pela remoção de vértices.

Lewis e Yannakakis (1980) apresentaram uma prova genérica sobre problemas de remoção de vértices, mostrando que qualquer problema deste tipo em grafos orientados (ou não orientados) para uma propriedade Π hereditária e não trivial é NP-Difícil. Π é considerada não trivial se é verdadeira para uma infinidade de grafos e falsa para uma infinidade de grafos, enquanto Π é hereditária se, para qualquer grafo G que satisfaça Π , todo subgrafo induzido por vértices de G também satisfaz Π . Notadamente ser livre de garras é uma propriedade não trivial e hereditária. Portanto, o Problema de Eliminação de Garras (PEG) é NP-Difícil.

Este trabalho está voltado para o Problema da Eliminação de Vértices (PEV, do inglês, *Vertex Deletion Problem*), visando a eliminação de garras para um dado grafo G de entrada, de forma a tornar G um grafo livre de garras. Desta forma, o PEG busca encontrar o menor conjunto $S \subseteq V(G)$, de forma que, $G[V(G) \setminus S]$ seja livre de garras.

A Figura 1(a) ilustra um grafo bipartido completo $K_{1,3}$. A Figura 1(b) exibe um grafo de entrada que possui uma garra ($G[\{s, a, b, c\}]$) como subgrafo induzido por vértices. Já a figura 1(c) mostra uma possível remoção de vértice de G que resulta em um grafo livre de garras.

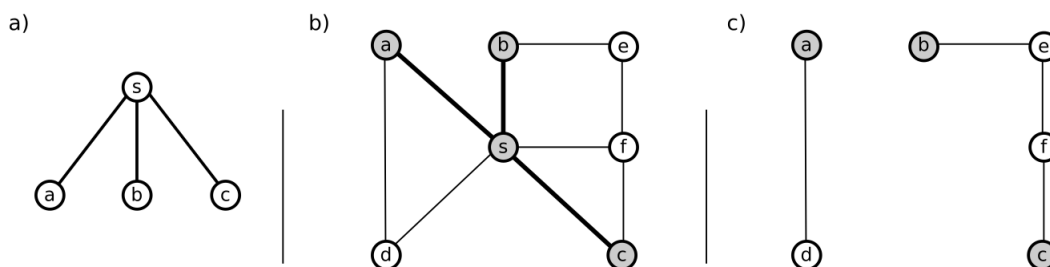


Figura 1: (a) Grafo bipartido completo $K_{1,3}$ (Garra). (b) Exemplo de uma Garra em um grafo G . (c) Grafo G' livre de Garras.

Desta maneira, foi proposto dois métodos, baseados, respectivamente, nas meta-heurísticas VNS e Algoritmos Genéticos, para a resolução do PEG. Ambos utilizam o conceito de busca local em vizinhança para se aproximar gradativamente do melhor resultado possível. O primeiro consiste em explorar vizinhanças cada vez mais distantes da solução corrente, focalizando a busca em torno

de uma nova solução se, e somente se, um movimento de melhora é executado. No segundo, um conjunto de soluções é criado aleatoriamente formando uma primeira população, que então passa repetidamente por processos de combinação e seleção, análogos aos da natureza. A aplicação destes processos faz com que a população tenda, iterativamente, a se aproximar de uma solução cada vez mais apta (melhor). Por fim, os resultados são comparados com as soluções ótimas obtidas por Fragoso (2019), por meio de Programação Linear Inteira (PLI).

O restante deste trabalho está organizado como segue: a Seção 2 discute alguns trabalhos relacionados ao tema; a Seção 3 apresenta os métodos propostos; os experimentos computacionais são discutidos na Seção 4; enquanto que as considerações finais e propostas de trabalhos futuros são trazidos na Seção 5.

2. Trabalhos relacionados

Grafos livres de garras e suas características têm sido abordados por diversos autores ao longo das últimas décadas. Dentre os trabalhos da área podemos citar Minty (1980), que tem como um dos objetivos do artigo, o de fornecer uma construção eficiente para um Conjunto Independente Máximo em grafos livres de garras; Wen-Lian e Nemhauser (1982), que apresenta um algoritmo em tempo polinomial para a Cobertura por Cliques de Peso Mínimo, computacionalmente mais prático que os algoritmos previamente apresentados por outros autores; e o Problema de Conversão de Grafos de Intervalo em Grafos de Intervalo Unitário, com contribuições variadas, estudadas por Fishburn (1985), Roberts (1979) e Bogart e West (1999).

Também podemos citar os estudos específicos realizados por Faudree et al. (1997), sobre a família dos Grafos Livres de Garras e suas diversas características; e o trabalho de Sales e Maffray (1998), que apresenta um algoritmo em tempo polinomial para a distinção de Grafos Livres de Garras nas classes "*quasi-parity*" e "*perfectly contractile*". Além da menção do Problema da Geração de Grafos de Intervalo Unitário por Eliminação Mínima de Vértices (Bevern et al., 2010), essencial para o PEG, explorado no presente trabalho; e o estudo poliédrico do politopo associado ao Problema Livre de Garras, discutido por Fragoso et al. (2019), com a apresentação de duas famílias de facetas, novas inequações de garra e estrela, além de um algoritmo de Planos de Corte com heurísticas de separação em instâncias randômicas de grafos, com grande desempenho das inequações do tipo estrela.

Um estudo poliédrico do politopo associado ao Problema de Eliminação de Garras também é apresentado por Fragoso (2019), explicitando sua dimensionalidade e revelando inequações definidoras de facetas, com instâncias intervalares que foram utilizadas para execução de testes computacionais para quatro algoritmos *Branch-and-Cut* e dois algoritmos *Branch-and-Price* propostos. Estas instâncias, por sua vez, foram utilizadas neste artigo, para fins de comparação de resultados.

3. Métodos propostos

Nesta Seção, são apresentados os métodos propostos para o PEG. Foram aplicadas abordagens baseadas nas meta-heurísticas VNS e Algoritmos Genéticos para a remoção de garras. Em um primeiro momento apresentamos estruturas e algoritmos utilizados em ambos os métodos, como a busca pela existência de garras, e a Busca Local.

3.1. Busca por Garras

Uma Lista de Garras previamente computada é utilizada para facilitar os dois processos de busca por garras que foram implementados nos métodos. A partir da lista de vértices vizinhos de cada vértice, $N(v)$, tomam-se os vizinhos de um vértice 3 a 3 e verifica-se a existência de arestas

entre eles. Desta forma, se as ligações entre estes 3 vizinhos forem inexistentes, tem-se uma garra no grafo G . Esta é, então, adicionada à Lista de Garras com centro no vértice v .

A busca pela existência de garras, passo recorrente em ambos os algoritmos apresentados, foi implementada e utilizada de duas maneiras diferentes. A primeira é uma busca geral por todos os vértices v que compõem a solução candidata S , bastando verificar se alguma garra com centro em v existe, essa implementação pode ser observada no Algoritmo 1. A segunda forma, observada no Algoritmo 2, é utilizada sempre que um vértice é adicionado à solução corrente, a ideia é que dado um subgrafo livre de garras $G[V']$, o subgrafo resultante $G[V' \cup \{u\}]$ só terá garras com centro em u ou em um dos vizinhos de u , bastando verificar se nenhuma das garras com centro em um desses vértices existe.

Algoritmo 1: Busca de Garras

```
1 Procedimento Busca de Garras ( $S$ )
2   Para cada vértice  $v \in S$  faça
3     Seja  $C(v)$  o conjunto de garras com o vértice  $v$  central;
4     Para cada  $c \in C(v)$  faça
5       Para cada  $v1, v2, v3 \in c$  faça
6         Se  $v1, v2$  e  $v3 \in S$  então
7           Retorne verdadeiro;
8   Retorne falso;
```

Algoritmo 2: Busca de Garras Minimizada

```
1 Procedimento Busca de Garras Minimizada ( $S, v$ )
2   Seja  $N(v)$  o conjunto de vizinhos do vértice  $v$ , incluindo  $v$ ;
3   Para cada vértice  $u \in N(v)$  faça
4     Seja  $C(u)$  o conjunto de garras com o vértice  $u$  central;
5     Para cada  $c \in C(u)$  faça
6       Para cada  $v1, v2, v3 \in c$  faça
7         Se  $v1, v2$  e  $v3 \in S$  então
8           Retorne verdadeiro;
9   Retorne falso;
```

3.2. Busca Local

A função proposta para realizar a busca local, detalhada no Algoritmo 3, é baseada no método da descida em vizinhança, que procura o melhor vizinho da solução corrente S . O algoritmo inicia a busca retirando todas as garras da solução e, após isso, percorre a vizinhança que é definida pelo conjunto de soluções que divergem em apenas um vértice da S e retorna a melhor entre elas.

3.3. Algoritmo Genético

Segundo Holland (1992), os pesquisadores perceberam o poder dos mecanismos evolutivos baseados na teoria da evolução de Darwin como algo que pode ser imitado e aplicado nas mais diversas áreas, inclusive no desenvolvimento de algoritmos. Nesta área, programas que simulassem a evolução poderiam ter capacidade de gerar soluções para problemas em que as estruturas não fossem totalmente compreendidas.

Algoritmo 3: Busca Local

```
1 Procedimento Busca-Local ( $S, G = (V, E)$ )
2   Enquanto  $S$  tem garra faça
3     remova aleatoriamente uma garra de  $S$ ;
4    $\hat{V} \leftarrow V \setminus S$ ;
5   Enquanto  $\hat{V}$  não vazio faça
6      $u \leftarrow$  vértice aleatório de  $\hat{V}$ ;
7      $S \cup \{u\} \leftarrow S \cup \{u\}$ ;
8     Se  $S \cup \{u\}$  livre de garras então
9        $S \leftarrow S \cup \{u\}$ ;
10     $\hat{V} \leftarrow \hat{V} \setminus \{u\}$ ;
11  Retorne  $S$ ;
```

Uma das técnicas desenvolvidas baseando-se nos processos evolutivos foram os Algoritmos Genéticos (AG), um tipo de meta-heurística desenvolvida por Holland na década de 1970. Nesses algoritmos, um conjunto de soluções é criada para formar uma população, que passa repetidamente por processos evolutivos, que em teoria, gerarão uma população mais apta. Dois dos mecanismos utilizados diretamente pelos Algoritmos Genéticos são: a reprodução e a seleção populacional; o primeiro é responsável por combinar as soluções para criar os novos membros da população e o segundo é responsável por selecionar quais soluções sobreviverão para a formar as próximas gerações. A proposta de Algoritmo Genético utilizada neste trabalho está descrita no Algoritmo 4.

Algoritmo 4: Algoritmo Genético

```
1 Procedimento AG ( $G = (V, E), \text{TempoLimite}$ )
2    $P \leftarrow$  Gerar-População ( $G = (V, E)$ );
3   Enquanto Não Atingir o TempoLimite faça
4      $P' \leftarrow$  Reproduzir População ( $P, G = (V, E)$ );
5      $P \leftarrow$  Selecionar População ( $P', G = (V, E)$ );
6   Retorne  $P$ ;
```

O passo inicial do Algoritmo Genético é a criação da primeira população, a função que executa esse processo é detalhada no Algoritmo 5. Nesta, é gerado um conjunto de soluções sem garras. Para criar um indivíduo com essa característica, sempre se inicia com uma solução S vazia e adiciona-se vértices aleatoriamente. Caso essa modificação não gere nenhuma garra, esta se torna permanente, caso contrário basta desfazê-la. Após realizar esse processo para todos os vértices disponíveis, tem-se uma solução inicial livre de garras.

A função que executa a seleção da população, detalhada no Algoritmo 6, cria uma nova população e a preenche em 80% com os melhores indivíduos da população anterior, garantindo que as melhores características sejam preservadas. Os 20% restantes são preenchidos por soluções aleatórias. Isto ajuda a aumentar a probabilidade de fuga dos ótimos locais, sendo uma alternativa à mutação, já que sempre serão adicionados indivíduos com novas características.

Por sua vez, a função que executa a reprodução é descrita no Algoritmo 7. Nesta, aplicando o elitismo, a população é dividida estrategicamente em dois grupos. O primeiro, dito elite,

Algoritmo 5: Gerar população

```

1 Procedimento Gerar-População ( $G = (V, E)$ )
2   Seja  $P \leftarrow \emptyset$  o conjunto das soluções iniciais;
3   para  $i$  de 1 até  $|V|$  faça
4     Seja  $S \leftarrow \emptyset$  uma solução inicial sem vértices;
5      $\hat{V} \leftarrow V$ ;
6     para  $j$  de 1 até  $|V|$  faça
7        $u \leftarrow$  vértice aleatório de  $\hat{V}$ ;
8        $S \cup \{u\} \leftarrow S \cup \{u\}$ ;
9       Se  $S \cup \{u\}$  livre de garras então
10         $S \leftarrow S \cup \{u\}$ ;
11         $\hat{V} \leftarrow \hat{V} \setminus \{u\}$ ;
12       $P \leftarrow P \cup \{S\}$ ;
13   Retorne  $P$ ;
```

Algoritmo 6: Selecionar População

```

1 Procedimento Selecionar População ( $P, G = (V, E)$ )
2    $P' \leftarrow |V| \times 0.80$  soluções mais aptas de  $P$ ;
3    $P_A \leftarrow$  Gerar População( $G = (V, E), |V| \times 0.20$ );
4    $P' \leftarrow P' \cup P_A$ ;
5   Retorne  $P'$ ;
```

é formado pelos 25% indivíduos mais aptos e o segundo pelo restante, soluções menos aptas da população. Os membros da elite se reproduzem apenas com os membros de fora da elite com objetivo de minimizar o surgimento de características dominantes que aprisionem a população em ótimos locais. O processo de reprodução ocorre sempre a partir de pares, cada dupla de soluções é combinada utilizando uma estratégia de recombinação por caminhos (*Path-relinking*), gerando outras duas soluções. No final do processo, o tamanho da população é dobrado, o que é desfeito durante a etapa de seleção.

Segundo Resende e Ribeiro (2005), *Path-relinking* é uma estratégia de intensificação que explora as trajetórias que conectam soluções para encontrar outras melhores. Na aplicação básica dessa estratégia, parte-se de uma solução S_1 e é realizada uma série de modificações para torná-la igual a uma solução S_2 , onde a cada modificação uma solução intermediária é gerada. A proposta de *Path-relinking* utilizada neste trabalho, descrita no Algoritmo 8, utiliza como modificação a retirada/adição de um vértice que pertence ao conjunto da diferença simétrica de S_1 e S_2 . Para melhorar os resultados encontrados, definiu-se que todas as soluções intermediárias passariam por uma busca local e para evitar que o desempenho fosse muito prejudicado, o número de modificações executadas foi limitada a quatro.

3.4. Meta-heurística VNS

Proposta por Mladenović e Hansen (1997), a Busca em Vizinhança Variável (VNS, do inglês *Variable Neighborhood Search*), é um método de busca local que explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança. Diferentemente de outras meta-heurísticas que se baseiam em métodos de busca local, o método VNS não segue uma trajetória única, mas explora vizinhanças cada vez mais distantes da solução corrente e focaliza a busca em torno de uma

Algoritmo 7: Reproduzir População

```

1 Procedimento Reproduzir População ( $P, G = (V, E)$ )
2   Seja  $P_E \leftarrow$  Os 25% mais aptos de  $P$ ;
3   Seja  $P_R \leftarrow P \setminus P_E$ , o restante da população;
4   Enquanto  $P_R \neq \emptyset$  faça
5     Se  $P_E \neq \emptyset$  então
6       Seja  $A$  uma solução aleatória de  $P_E$ ;
7        $P_E \leftarrow P_E \setminus A$ ;
8     senão
9       Seja  $A$  uma solução aleatória de  $P_R$ ;
10       $P_R \leftarrow P_R \setminus A$ ;
11      Seja  $B$  uma solução aleatória de  $P_R$ ;
12       $P_R \leftarrow P_R \setminus B$ ;
13       $P \leftarrow P \cup \{\text{Path-Relinking}(A, B), \text{Path-Relinking}(B, A)\}$ ;
14   Retorne  $P$ ;
```

Algoritmo 8: Path-Relinking

```

1 Procedimento Path-Relinking ( $S_1, S_2, G = (V, E)$ )
2   Seja  $\hat{V} \leftarrow S_1 \Delta S_2$ , a diferença simétrica entre  $S_1$  e  $S_2$ ;
3   Seja  $M \leftarrow S_1$ , a solução filha mais apta;
4   Seja  $I \leftarrow S_2$ , uma solução intermediária;
5   Seja  $B$  uma solução encontrada a partir de uma busca local na solução intermediária;
6   Para cada  $j$  de 1 até 4 faça
7      $u \leftarrow$  Vértice aleatório de  $\hat{V}$ ;
8     Se  $u \in I$  então
9        $I \leftarrow I \setminus \{u\}$ ;
10    senão
11       $I \leftarrow I \cup \{u\}$ ;
12     $B \leftarrow$  Busca-Local ( $A, G = (V, E)$ );
13    Se  $B$  é mais apto que  $M$  então
14       $M \leftarrow B$ ;
15     $\hat{V} \leftarrow \hat{V} \setminus \{u\}$ ;
16   Retorne  $M$ ;
```

nova solução se, e somente se, um movimento de melhora é executado. O método também engloba um procedimento de busca local aplicado sobre a solução corrente.

Assim, foi apresentado um método baseado na Meta-heurística VNS. Aqui partimos sempre de uma solução inicial qualquer, obtida utilizando uma construção aleatória, com uma certa repetição, que retorna a melhor dentre as soluções geradas, como demonstra o Algoritmo 9.

Algoritmo 9: Solução Inicial Aleatória para VNS

```

1  Procedimento solução_inicial_aleatória( $G = (V, E)$ )
2     $S^* \leftarrow V$ ;
3    para  $i$  de 0 até 10 faça
4       $V' \leftarrow V$ ;
5      Seja  $S \leftarrow \emptyset$  uma solução inicial sem vértices;
6      Enquanto  $V'$  não vazio faça
7         $u \leftarrow$  vértice aleatório de  $V'$ ;
8         $S \cup \{u\} \leftarrow S \cup \{u\}$ ;
9        Se  $S \cup \{u\}$  livre de garras então
10          $S \leftarrow S \cup \{u\}$ ;
11         $V' \leftarrow V' \setminus \{u\}$ ;
12      Se  $S$  melhor que  $S^*$  então
13         $S^* \leftarrow S$ ;
14    Retorne  $S^*$ ;

```

Partindo desta solução inicial S , a cada iteração, seleciona-se de forma aleatória um vizinho $S \cup \{u\}$ dentro da vizinhança da solução corrente. Esse vizinho é submetido a um procedimento de busca local, como mostrado na Seção 3.1 - retornando uma solução ótima local, S'' . Se S'' for melhor que a solução corrente, a busca continua de S'' . Caso contrário, a solução corrente passa por um processo de perturbação, no qual, gradativamente, uma quantidade crescente de vértices - b - é aleatoriamente removida ou adicionada à solução, como artifício da meta-heurística para fugir de ótimos locais. Encerra-se o procedimento quando uma condição de parada é atingida, no nosso caso, um tempo máximo permitido de CPU. O Algoritmo 10 apresenta estes procedimentos. O método de busca local pode ser visto no Algoritmo 3.

4. Experimentos computacionais

Os métodos estudados neste trabalho foram desenvolvidos em linguagem C++, utilizando o compilador g++ versão 6.3.0 (com opção de compilação -O3). Os experimentos computacionais foram executados em um computador com processador Intel Core i5-8265U (64 bits), com CPU contendo 8 cores de 1,60 GHz, além de 7,6 GiB de RAM e executando o sistema operacional Linux Ubuntu 19.10.

Para os experimentos realizados, grafos de intervalo gerados de forma aleatória por Frago (2019) foram utilizados. Foram utilizados 27 grupos de instâncias de grafos intervalares foram criados combinando os parâmetros $n \in \{100, 200, 300\}$, tempo total de intervalo = 100, e comprimento máximo $\in \{10, 20, 30, 40, 50, 60, 70, 80, 90\}$. Cada subgrupo é definido pela densidade de suas nove instâncias, totalizando 243 grafos-problema.

Além disso, foram utilizadas as instâncias aleatórias de grafos geradas por Bastos et al. (2016) com $n \in \{50, 100\}$. Cada grupo é composto por 10 instâncias, sendo dois grupos para $n = 50$ e dois grupos para $n = 100$, totalizando 40 grafos-problema.

Algoritmo 10: Meta-heurística VNS para o PEG

```

1  Procedimento VNS ( $G = (V, E)$ ,  $tempoLimite$ )
2     $S \leftarrow \text{solução\_inicial\_aleatoria}(G)$ ;
3    Enquanto não atingir  $tempoLimite$  faça
4       $b \leftarrow 0$ ;
5       $b_{max} \leftarrow |V|$ ;
6      Enquanto ( $b < b_{max}$ ) E (não atingir  $tempoLimite$ ) faça
7         $S \cup \{u\} \leftarrow \text{perturbação}(S, b)$ ;
8         $b \leftarrow b + 1$ ;
9         $S \cup \{u\} \leftarrow \text{busca-local}(S \cup \{u\})$ ;
10       Se  $S \cup \{u\}$  melhor que  $S$  então
11          $S \leftarrow S \cup \{u\}$ ;
12          $b \leftarrow 1$ ;
13      $S \cup \{u\} \leftarrow \text{solução\_inicial\_aleatoria}(|V|)$ ;
14  Retorne  $S$ ;

```

Os métodos baseados nas meta-heurísticas VNS e Algoritmos Genéticos foram executados 10 vezes para cada instância, visto que estes métodos possuem componentes aleatórios. Ainda, foram utilizados tempos limites para execução dos testes, proporcionais à quantidade de vértices destas instâncias. Para as instâncias com $|V| \in \{50, 100, 200, 300\}$, temos 10, 30, 60 e 120 segundos de limite, respectivamente. A comparação dos resultados computacionais dos métodos propostos neste trabalho estão explicitadas nas Tabelas de 1 a 3, onde cada linha está representando um dos grupos de nove instâncias de dimensão igual, dos grafos intervalares; e na Tabela 4, cada linha representando um dos grupos de 10 instâncias de dimensão igual, dos grafos aleatórios.

Nestas tabelas pode-se observar as duas primeiras colunas representando os grupos de instâncias testados, seguidos da média de remoções ótimas de vértices (OPT), encontradas por Fragoso (2019) em seu trabalho. Nas linhas em que o OPT está acompanhado de um “*”, têm-se que nem todas as instâncias do grupo têm um número exato de remoções ótimas de vértices encontradas por Fragoso. Nestes casos, a média de OPT foi obtida mesclando a média de OPT das instâncias que estavam disponíveis com a média dos melhores resultados encontrados pelos métodos propostos neste trabalho.

As colunas seguintes representam os resultados dos métodos testados. Os grupos de colunas VNS e Genetic representam, respectivamente, os resultados do método baseado na meta-heurística VNS e do método baseado nos algoritmos genéticos. Além disso, ambos os métodos possuem a coluna Best e Worst representando o melhor e pior resultado da média das 9/10 instâncias, a coluna Av (average) para o resultado médio obtido após executar as 9/10 instâncias 10 vezes, e, por fim, a coluna Time para a média do tempo em segundos em que o melhor resultado foi obtido.

Pode ser observado na Tabela 1 que, em todas as instâncias intervalares de $|V| = 100$, ambos os métodos chegam ao número de remoções de vértices ótimo (OPT), com nenhuma diferença. O tempo retornado pelos métodos também é muito satisfatório. Tendo o VNS obtido os melhores tempos para todas as instâncias.

Na Tabela 2, de instâncias intervalares com $|V| = 200$, a média de tempos dos dois métodos se mostrou muito próxima. O método VNS consegue chegar ao número de remoções de vértices ótimo nas suas melhores execuções, com variações pequenas na média das 10 execuções. O método Genético também consegue chegar ao OPT na maioria das suas melhores execuções, além

Tabela 1: Resultados para instâncias intervalares com $|V| = 100$

Inst	D	OPT	VNS				Genetic			
			Best	Worst	Average	Time	Best	Worst	Average	Time
100	10	20,8	20,8	20,8	20,8	0,01217	20,8	20,8	20,8	0,25594
	20	31	31	31	31	0,20491	31	31	31	3,73656
	30	31,1	31,1	31,1	31,1	0,13742	31,1	31,1	31,1	2,52286
	40	31,6	31,6	31,6	31,6	0,19755	31,6	31,6	31,6	1,86190
	50	21,5	21,5	21,5	21,5	0,09649	21,5	21,5	21,5	1,05236
	60	11,2	11,2	11,2	11,2	0,02299	11,2	11,2	11,2	0,87554
	70	6,7	6,7	6,7	6,7	0,01712	6,7	6,7	6,7	0,51530
	80	3,8	3,8	3,8	3,8	0,01574	3,8	3,8	3,8	0,20446
	90	0,7	0,7	0,7	0,7	0,01408	0,7	0,7	0,7	0,03965

de obter resultados próximos do ótimo nas demais com uma diferença máxima de 2.0. A média de tempos, em ambos os métodos, foi excelente - mesmo as médias para os três primeiros grupos de instâncias no método genético estando acima do método VNS, ainda representam um baixo esforço computacional.

Tabela 2: Resultados para instâncias intervalares com $|V| = 200$

Inst	D	OPT	VNS				Genetic			
			Best	Worst	Average	Time	Best	Worst	Average	Time
200	10	58,5	58,5	58,8	58,7	4,7735	59,4	61,8	60,6	26,8873
	20	72,8*	72,8	73,5	73,2	6,8276	74,8	80,1	77,5	47,6010
	30	70,5*	70,6	70,6	70,6	4,7712	71,1	77	74,1	14,5153
	40	67,5	67,5	67,6	67,6	9,0264	67,5	69,7	68,6	7,6852
	50	44,8	44,8	44,8	44,8	4,4423	44,8	44,8	44,8	3,7704
	60	25,2	25,2	25,2	25,2	1,5338	25,2	25,2	25,2	1,0407
	70	15,2	15,2	15,2	15,2	0,4809	15,2	15,2	15,2	0,3659
	80	9,2	9,2	9,2	9,2	0,2309	9,2	9,2	9,2	0,1099
	90	2,3	2,3	2,3	2,3	0,1840	2,3	2,3	2,3	0,0305

Na Tabela 3, é possível observar que o mesmo padrão anterior ocorre para as instâncias intervalares de $|V| = 300$. Com a diferença de que agora temos uma diferença máxima entre as melhores execuções e o OPT de 6.1. O tempo médio retornado pelo método Genético também fica melhor que o tomado pelo método VNS na maioria das vezes.

Por fim, ao analisar os resultados dos grafos aleatórios com $|V| = 50$ e $|V| = 100$ da Tabela 4, também foi possível perceber que os resultados alcançados ficaram muito próximos aos valores ótimos, e desta vez possuem uma diferença máxima de 2.0 na suas médias de execução. O tempo levado por ambos os métodos foi muito satisfatório, em vista da complexidade maior dos grafos em questão, quando comparados às instâncias intervalares.

5. Considerações finais e proposta de trabalhos futuros

Neste trabalho propomos dois métodos baseados em meta-heurísticas para o Problema de Eliminação de Garras. O primeiro método, baseado na meta-heurística Algoritmos Genéticos

Tabela 3: Resultados para instâncias intervalares com $|V| = 300$

Inst	D	OPT	VNS				Genetic			
			Best	Worst	Average	Time	Best	Worst	Average	Time
300	10	103,5*	103,5	104,8	104,2	27,45	109,5	115,1	112,3	98,70
	20	120,5*	120,5	122,8	121,7	24,33	126,6	136,5	131,6	23,01
	30	111,1*	111,1	112,3	111,7	25,07	113,4	121,4	117,4	8,87
	40	98*	98	98,3	98,1	73,67	98	101,4	99,7	40,25
	50	69,8	69,8	69,8	69,8	31,83	69,8	69,8	69,8	27,512
	60	37,6	37,6	37,6	37,6	10,93	37,6	37,6	37,6	8,80
	70	24,4	24,4	24,4	24,4	5,64	24,4	24,4	24,4	5,33
	80	14,4	14,4	14,4	14,4	2,11	14,4	14,4	14,4	1,31
	90	4,5	4,5	4,5	4,5	0,93	4,5	4,5	4,5	0,16

Tabela 4: Resultados para instâncias aleatórias com $|V| \in \{50, 100\}$

Inst	D	OPT	VNS				Genetic			
			Best	Worst	Average	Time	Best	Worst	Average	Time
50	105	27,6*	27,6	27,6	27,6	1,46772	27,6	31	29,3	0,00475
50	106	28,7	28,7	28,7	28,7	0,87439	29,9	31,4	30,6	0,00229
100	109	69,7*	69,7	71,2	70,4	8,25592	70,6	72,8	71,7	12,46072
100	110	72,9*	72,9	74,7	73,8	6,77151	73,3	75,2	74,2	14,42478

e o segundo na meta-heurística VNS. A primeira meta-heurística, Genética, procura encontrar a melhor solução aplicando um mecanismo de reprodução, baseado nos processos evolutivos, sobre uma população de soluções, encontrando o “indivíduo mais apto” - mais próximo às características impostas. Já a segunda meta-heurística, VNS, procura vizinhos gradativamente mais afastados da solução corrente, construindo a solução a partir de uma solução inicial aleatória, com melhorias de busca local.

Foram executados testes computacionais os quais demonstraram que os métodos propostos foram capazes de encontrar soluções de alta qualidade com pequeno esforço computacional, em todas as instâncias intervalares, assim como nas instâncias randômicas. Esta comparação também foi possível graças a Fragoso (2019) em seu trabalho, a partir de seus valores ótimos para remoção de vértices.

Como planejamentos de trabalhos futuros, pretendemos melhorar as estruturas de dados, como forma de aprimorar ainda mais o desempenho dos dois algoritmos, implementando também o conceito de remoção de estrelas, apresentado no trabalho de Fragoso (2019). Além disso, pretendemos investigar novas abordagens de heurísticas para o problema, tais como Colônia de Formigas e GRASP.

Referências

- Bastos, L., Ochi, L. S., Protti, F., Martins, I. C., Pinheiro, R. G. S., e Subramanian, A. (2016). Efficient algorithms for cluster editing. *Journal of Combinatorial Optimization*, 31:347–371.
- Bevern, R. v., Komusiewicz, C., Moser, H., e Niedermeier, R. (2010). Measuring indifference: Unit interval vertex deletion. *Graph Theoretic Concepts in Computer Science. Lecture Notes in Computer Science*, 6410:232–243.
- Bogart, K. P. e West, D. B. (1999). A short proof that proper = unit. *Discrete Mathematics*, 201(1): 21–23. ISSN 0012-365X.
- Faudree, R., Flandrin, E., e Ryjáček, Z. (1997). Claw-free graphs—a survey. *Discrete Mathematics*, 164(1-3):87–147.
- Fishburn, P. C. (1985). Interval graphs and interval orders. *Discrete Applied Mathematics*, 55(2): 135–149.
- Fragoso, F. C. (2019). Graph declawing problem: Polyhedra and exact solutions. Master thesis submitted for the programme of msc on computer science, Universidade Federal da Paraíba.
- Fragoso, F. C., Gilberto, F. d. S. F., e Protti, F. (2019). Algoritmos de planos de corte para o problema livre de garra. In *Anais do LI Simpósio Brasileiro de Pesquisa Operacional*.
- Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1):66–73.
- Lewis, J. M. e Yannakakis, M. (1980). The node-deletion problem for hereditary properties is np-complete. *Journal of Computer and System Sciences*, 20(2):219–230.
- Minty, G. J. (1980). On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory, Series B*, 28(3):284–304.
- Mladenović, M. e Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100.
- Resende, M. G. C. e Ribeiro, C. C. (2005). Grasp with path-relinking: Recent advances and applications. In *Metaheuristics: progress as real problem solvers*, p. 29–63. Springer.
- Roberts, F. S. (1979). Indifference and seriation. *Annals of the New York, Academy of Sciences*, 328 (1):173–182. ISSN 1749-6632.
- Sales, C. L. e Maffray, F. (1998). Even pairs in claw-free perfect graphs. *Journal of Combinatorial Theory, Series B*, 74(2):169–191.
- Wen-Lian, H. e Nemhauser, G. L. (1982). A polynomial algorithm for the minimum weighted clique cover problem on claw-free perfect graphs. *Discrete Mathematics*, 38(1):65–71.