# Uma abordagem algorítmica para teoria algébrica de grafos

**Nome do Aluno:** Leonardo Bertucci dos Santos

**RA do aluno:** 11028714

**E-mail do aluno:** leonardo.bertucci@aluno.ufabc.edu.br

**Nome do orientador:** Cristiane Maria Sato

**E-mail do orientador:** c.sato@ufabc.edu.br

**Palavras-chave do projeto:** grafos, teoria algébrica dos grafos, core, homomorfismos

**Área de conhecimento do projeto:** Ciência da Computação

# 1. Conceitos iniciais

Neste capítulo apresentamos as definições básicas a respeitos de grafos e homomorfismos de grafos, mostrando alguns resultados iniciais e algoritmos dentro do tema.

## 1.1. Grafos

Um **grafo** $G$ consiste de um conjunto de **vértices** $V(G)$ e um conjunto de **arestas** $E(G)$ tal que uma aresta é um par não ordenado de elementos distintos de $V(G)$. Denotamos uma aresta $(u, v)$ simplesmente por $uv$. Se $e = uv$ é uma aresta de $G$ dizemos que $u$ e $v$ são **adjacentes** ou **vizinhos**, e escrevemos $u \sim v$; dizemos também que $e$ **liga** os vértices $u$ e $v$, e que **incide** sobre cada um deles. O **grau** de um vértice $v$, $d(v)$, é o número de arestas incidentes a ele, e a **vizinhança** de $v$ o conjunto contendo seus vértices vizinhos, denotada $N(v)$. Um vértice que possui grau zero é dito **isolado**. A **ordem** ou **tamanho** de um grafo $G$ é o seu número de vértices. Um grafo é dito **completo** se todos os seus vértices são adjacentes, sendo denotado $K_n$ o grafo completo de ordem $n$, e dito **vazio** se não possui nenhuma aresta (mas pelo menos um vértice). O grafo sem vértices nem arestas é chamado **grafo nulo**.

Grafos da forma que definimos aqui são chamados de **grafos simples**, pois existem algumas definições mais gerais que permitem por exemplo arestas paralelas ou loops (aresta de um vértice a si mesmo). Uma generalização importante ocorre se considerarmos pares ordenados de $V(G)$, denominados **arcos** ou **arestas dirigidas**, no lugar das arestas. Definimos desta forma um **grafo dirigido**, ou **digrafo**, como $V(G)$ junto com um conjunto de arcos $A(G)$. Podemos ver nesse contexto um grafo simples como um grafo dirigido onde $(v, u)$ é um arco sempre que $(u, v)$ for um arco. Mencionaremos neste texto sempre que formos trabalhar com digrafos ou qualquer outra variação de grafo, e caso contrário usaremos a palavra "grafo" sempre para identificar um grafo simples com conjunto de vértices finito.

O **complemento** de um grafo $G$, denotado $\overline{G}$, é o grafo que possui o mesmo conjunto de vértices de $G$, com $u \sim v$ em $\overline{G}$ se e somente se $u \nsim v$ em $G$. Um **clique** em um grafo $G$ é um conjunto de vértices mutualmente adjacentes, enquanto um **conjunto independente/estável** é um conjunto de vértices mutualmente não-adjacentes. Pela definição de complemento pode-se ver que um clique de $G$ é um conjunto independente em $\overline{G}$, e vice-versa. Denotamos o tamanho de um clique máximo em $G$ por $\omega(G)$, chamado **número de clique** de $G$, e um conjunto independente máximo por $\alpha(G)$, o **número de independência/estabilidade** de $G$.

## 1.2. Subgrafos

Um **subgrafo** de um grafo $G$ é um grafo $H$ tal que

$$V(H) \subseteq V(G), \quad E(H) \subseteq E(G).$$

Se $V(H) = V(G)$, dizemos que $H$ é **subgrafo gerador** de $G$; se $V(H) \neq V(G)$ então $H$ é um **subgrafo próprio**. $H$ é um **subgrafo induzido** se dois vértices em $V(H)$ são adjacentes se e somente se eles são adjacentes em $V(G)$. Um subgrafo gerador pode ser obtido deletando-se algumas arestas de $G$, enquanto um subgrafo induzido pode ser obtido ao se deletar alguns vértices de $G$ (junto com as arestas que se ligavam a eles).

Dado um subconjunto $S \subseteq V(G)$, o **subgrafo induzido por** $S$, denotado $G[S]$, é o subgrafo induzido de $G$ que possui como vértices o conjunto $S$. Se $F \subseteq E(G)$, $G[F]$ é o **subgrafo induzido por** $F$, com conjunto de vértices dado por $V(G[F]) = \{u \in V(G) : uv \in F\}$ e conjunto de arestas $E(G[F]) = F$.

Um **passeio** é uma sequência de vértices $(v_0, v_1, \ldots, , v_k)$ tal que $v_i \sim v_{i+1}$ para todo $0 \leq i \leq k$; seu **comprimento** é o número de arestas que possui, e dizemos que é **fechado** quando $v_0 = v_k$. Poderemos tratar um passeio $w$ como o subgrafo induzido por suas arestas quando conveniente. Uma **trilha** é um passeio que não repete arestas, enquanto um **caminho** é um passeio que não repete vértices. $P_k$ denota o caminho de comprimento $k$. Um **ciclo** é um passeio fechado que não repete vértices, com excessão dos extremos, sendo $C_n$ o ciclo de tamanho $n$.

Um grafo $G$ é **bipartido** se seu conjunto de vértices pode ser particionado em dois conjunto $A$ e $B$ de modo que toda aresta de $G$ liga somente vértices entre $A$ e $B$.

**Teorema 1.** *Um grafo é bipartido se e somente se não contém ciclos ímpares.*

*Demonstração.* Considere um ciclo ímpar $C_n = (1, 2, \ldots, n, 1)$, onde $n$ é um inteiro ímpar positivo. Suponha que exista bipartição $A, B$ de $C_n$, e que o vértice 1 pertenca a $A$. Como $i \sim i + 1$ para todo vértice $i$ de $C_n$, temos que todos os vértices pares estarão em $B$ e os ímpares estarão em $A$. Logo $n \in A$. Mas $n$ é adjacente a 1, chegando a uma contradição.

Para a volta, vamos contruir uma bipartição $A, B$ de um grafo $G$ que não possua ciclos ímpares do seguinte modo: Tome $v \in V(G)$ e o adicione em $A$. Adicione então todos os vizinhos de $v$ ao conjunto $B$. Repita o processo de adicionar os vizinhos à outra partição (que já não estejam lá) até acabarem os vértices. Afirmamos que $A, B$ é uma bipartição de $G$, pois durante o processo, caso seja inserido um vértice $y$, por exemplo em $A$, que seja adjacente a um vértice $x$ que já está em $A$, então haveria um ciclo ímpar em $G$, formado pelo caminho de $x$ até $y$ que levou à inserção de $y$ em $A$ junto com a aresta $xy$. □

## 1.3. Homomorfismos

**Definição 1.** Sejam $G, H$ grafos. Uma função $f : V(G) \to V(H)$ é um **homomorfismo** se $f(u)$ e $f(v)$ são adjacentes sempre que $u$ é adjacente a $v$.

Um homomorfismo de $G$ em si mesmo é um **endomorfismo**. O conjunto de todos os endomorfismos em um grafo $G$ forma um monoide (estrutura algébrica com operação binária associativa e que possui elemento neutro). Se $f$ é um homomorfismo de $G$ em $H$, o grafo formado pelos vértices $\{f(u) : u \in V(G)\}$ e arestas $\{f(u)f(v) : uv \in E(G)\}$ é chamado **imagem homomórfica** de $G$ por $f$, e denotado $f(G)$. $f(G)$ é um subgrafo de $H$.

Uma propriedade interessante que pode ser caracterizada por meio de homomorfismos é o número cromático de um grafo: uma $k$-**coloração própria** de uma grafo $G$ é uma função de $V(G)$ em um conjunto de $k$ cores tal que vértices adjacentes são levados em cores diferentes. O menor número $k$ para o qual $G$ pode ser próprimente $k$-colorido é chamado **número cromático** de $G$, e é denotado por $\chi(G)$. O conjunto de vértices com uma determinada cor forma um conjunto independente.

**Lema 1.** *O número cromático de um grafo $G$, $\chi(G)$, é igual ao menor inteiro $r$ tal que existe um homomorfismo de $G$ para $K_r$.*

**Definição 2.** Uma **retração** é um homomorfismo de um grafo $G$ em um subgrafo $H$ de si mesmo tal que a restrição $f \restriction_H$ de $f$ para $V(H)$ é a identidade. Se existe uma retração de $G$ para um subgrafo $H$, dizemos também que $H$ **é uma retração de** $G$.

De fato, se existe $f : G \to H$ tal que $f \restriction_H$ seja uma bijeção, então $H$ será uma retração de $G$ via a função $g = (f \restriction_H)^{-1} \circ f$.

**Exemplo 1.** Subgrafo de $G$ que é imagem homomórfica mas não é retração (desenhar).

**Definição 3.** Um **isomorfismo** $\phi$ de $G$ em $H$ é um homomorfismo bijetivo cuja inversa é também um homomorfismo. Se $G$ e $H$ são isomorfos, escrevemos $G \cong H$.

Em outras palavras, $\phi$ é um isomorfismo quando $f(u)$ e $f(v)$ são adjacentes se e somente se $u$ e $v$ são adjacentes. Dois grafos isomorfos possuem exatamente a mesma estrutura e em geral podemos tratá-los como se fossem iguais.

Um isomorfismo de $G$ em si mesmo é chamado um **automorfismo**. O conjunto de todos os automorfismos em um grafo $G$ forma um grupo, denominado **grupo de automorfismos** de $G$ e denotado como $\mathrm{Aut}(G)$. O grupo de automorfismos de um grafo $G$ é um subgrupo do grupo simétrico $\mathrm{Sym}(V(G))$, o grupo de todas as permutações dos vértices de $G$. Se $G$ possui $n$ vértices, escreveremos $\mathrm{Sym}(n)$ ao invés de $\mathrm{Sym}(V(G))$. Em particular, $\mathrm{Aut}(K_n) \cong \mathrm{Sym}(n)$, já que toda permutação de vértices no grafo completo é um automorfismo.

**Proposição 1.** *$G$ e $\overline{G}$ possuem o mesmo grupo de automorfismos.*

*Demonstração.* Considere $\phi \in \mathrm{Aut}(G)$. Se $uv$ é aresta de $\overline{G}$, então $uv \notin E(G)$. Como $\phi$ é isomorfismo, $\phi(u)\phi(v) \notin E(G)$, e portanto $\phi(u)\phi(v)$ é aresta de $\overline{G}$. Em todos os passos utilizados vale a volta, logo $\phi$ é automorfismo de $\overline{G}$. □

Definimos uma relação $\to$ na classe de todos os grafos por $G \to H$ se existe homomorfismo de $G$ em $H$. Como a composição de homomorfismos é um homomorfismo,

$\to$ é transitiva; $\to$ é também reflexiva já que a identidade é um homomorfismo. Não é difícil ver que nossa nova relação não é simétrica nem anti-simétrica, e $\to$ é assim uma pré-ordem na classe de todos os grafos. Chamaremos $\to$ de **pré-ordem de homomorfismos**. Dois grafos que não admitem homomorfismo de um no outro são ditos **incomparáveis**, e caso contrário, são **comparáveis**. Dois grafos $G$ e $H$ tais que $G \to H$ e $H \to G$ são ditos **homomorficamente equivalentes**.

Se $f$ é um homomorfismo de $G$ para $H$, as pré imagens $f^{-1}(y)$ de cada vértice $y \in H$ determinam uma partição $\pi$ de $V(G)$ chamada **kernel de** $f$ e denotada por $\ker f$. O kernel de $f$ é uma partição em conjuntos independentes. Dado um grafo $G$ e uma partição $\pi$ de $V(G)$, definimos um grafo $G/\pi$ tomando as classes de $\pi$ como vértices e uma aresta entre duas classes se existe uma aresta em $G$ conectando estas classes. Existe um homomorfismo natural de $G$ em $G/\pi$ com kernel $\pi$. Note que $G/\pi$ será um grafo simples se e somente se $\pi$ for uma partição de $V(G)$ em conjuntos independentes.

**Teorema 2.** *Seja $f : G \to H$ homomorfismo. Então $G\big/\ker f \cong f(G)$.*

*Demonstração.* Defina

$$\begin{aligned} \phi : G\big/\ker f &\longrightarrow f(G) \\ f^{-1}(y) &\longmapsto y. \end{aligned}$$

A função $\phi$ é um isomorfismo, pois se $y_1, y_2$ são vértices de $f(G)$, então $f^{-1}(y_1) \sim f^{-1}(y_2) \iff \exists x_1 \in f^{-1}(y_1), x_2 \in f^{-1}(y_2) : x_1 \sim x_2 \iff y_1 \sim y_2$. □

**Corolário 1.** *Cada quociente de $G$ é uma imagem homomórfica de $G$, e por outro lado, cada imagem homomórfica de $G$ é isomorfa a um quociente de $G$.*

*Demonstração.* Dada uma partição $\pi$ de $G$, o homomorfismo natural $f_\pi$ que leva um elemento $x$ de $G$ em $[x]$ sua classe de equivalência satisfaz $f_\pi(G) = G\big/\pi$. Por outro lado, se $f$ é um homomorfismo de $G$, temos pelo teorema acima que $G\big/\ker f \cong f(G)$. □

**Definição 4.** Um grafo $G$ é um **core** (ou **núcleo**) se todo homomorfismo de $G$ em si mesmo é uma bijeção. Um subgrafo $H$ de $G$ é um **core de** $G$ se $H$ é um core e existe um homomorfismo de $G$ para $H$ ($G \to H$). Denotamos o core de $G$ por $G^\bullet$.

Podemos caracterizar cores como os grafos que possuem o monóide de endomorfismos igual ao seu grupo de automorfismos. O core de $G$ é uma retração minimal de $G$, com respeito a inclusão. O exemplo mais imediato de família de cores são os grafos completos $K_n$. Outro exemplo é dado pelos ciclos ímpares, que não podem possuir um homomorfismo para um subgrafo induzido como fica claro pela proposição a seguir:

**Proposição 2.** *A imagem homomórfica um ciclo ímpar de tamanho $n$ deve conter um ciclo ímpar de tamanho menor ou igual a $n$.*

*Demonstração.* Seja $C_n = (1, \cdots, n, 1)$ um ciclo ímpar e $f : C_n \to H$ um homomorfismo. A sequência $(f(1), \cdots, f(n), f(1))$ será um passeio fechado de tamanho $n$ em $f(C_n)$. Note no entanto que em um grafo bipartido todo passeio fechado deve ter tamanho par. Logo, a imagem de $f$ contém um ciclo ímpar. Claramente $f(C_n)$ possui ordem menor ou igual a $n$. □

Os cores formam uma classe de grafos em que a relação $\to$ é uma ordem parcial, a menos de isomorfismo. O lema a seguir demonstra a antissimetria de $\to$ neste conjunto:

**Lema 2.** *Sejam G e H cores. Então G e H são homomorficamente equivalentes se e somente se são isomorfos.*

*Demonstração.* Sejam $f : G \to H$ e $g : H \to G$ homomorfismos. Tanto $f \circ g$ quanto $g \circ f$ devem ser bijetivas já que $G$ e $H$ são cores. Portanto $f$ e $g$ também são bijetivas, e $G$ e $H$ são isomorfos. □

**Teorema 3.** *Todo grafo possui um core, que é um subgrafo induzido e único a menos de isomorfismo.*

*Demonstração.* Sendo $G$ um grafo finito e a identidade um homomorfismo, temos que o conjunto de subgrafos de $G$ que são a imagem de um homomorfismo de $G$ é finito, e portanto possui um elemento minimal. Sejam então $H_1$ e $H_2$ cores de $G$, imagens dos homomorfismos $f_1$ e $f_2$, respectivamente. Então $f_1 \restriction_{H_2}$ é homomorfismo de $H_2$ para $H_1$, assim como $f_2 \restriction H_1$ é homomorfismo de $H_1$ a $H_2$. Logo, pelo lema anterior, $H_1 \cong H_2$. Sendo um elemento minimal do conjunto descrito acima, o core de $G$ é uma retração, e portanto um subgrafo induzido. □

**Lema 3.** *Dois grafos G e H são comparáveis se e somente se seus cores são comparáveis.*

*Demonstração.* Se $f : G \to H$ é homomorfismo, então temos a sequência de homomorfismos

$$G^\bullet \to G \xrightarrow{f} H \to H^\bullet,$$

cuja composição mostra que $G^\bullet$ e $H^\bullet$ são comparáveis. Por outro lado, se $f : G^\bullet \to H^\bullet$ é homomorfismo, a sequência

$$G \to G^\bullet \xrightarrow{f} H^\bullet \to H$$

nos dá o homomorfismo entre $G$ e $H$. □

**Corolário 2.** *Dois grafos G e H são homomorficamente equivalentes se e somente se seus cores são isomorfos.*

*Demonstração.* Segue diretamente do lemma 3 e lemma 2. □

## 1.4. Algoritmos

Nesta seção exibiremos os algoritmos que foram desenvolvidos ao longo do trabalho acompanhando o estudo dos temas e poderemos discutir sobre a complexidade assintótica de alguns desses problemas, trazendo resultados que sejam relevantes. Os códigos foram feitos utilizando a linguagem python e utilizamos a representação matricial para representar grafos no computador.

Começamos verificando a existência de um homomorfismo de um grafo $G$ em um grafo $H$ e também se são isomorfos por força bruta, isto é, olhando para todas as funções dos vértices de $G$ nos vértices de $H$:

```
1  def verify_homo(G, H):
2      for f in gen_func(len(G), len(H)):
3          homo = True
4          # vejamos se f é homomorfismo:
5          for (i, j) in edges(G):
6              if H[f[i]][f[j]] == 0:
7                  homo = False
8                  break
9          if homo: return f
10     return False
```

```
1  def verify_iso(G, H):
2      if len(G) != len(H): return False
3      for f in list(itertools.permutations(list(range(len(G))))
           :
4          iso = True
5          # vejamos se f é homomorfismo:
6          for (i, j) in edges(G):
7              if H[f[i]][f[j]] == 0:
8                  iso = False
9                  break
10         if iso:  # vejamos se f^-1 é homomorfismo:
11             g = invert(f)
12             for (i, j) in edges(H):
13                 if G[g[i]][g[j]] == 0:
14                     return False
15         if iso: return f
16     return False
```

Aqui a função *gen_func* gera todas as funções dos vértices de $G$ nos vértices de $H$, possuindo complexidade $O(|V(H)|^{|V(G)|}$ e dando caráter exponencial ao algoritmo *verify_homo*. De fato, o caso geral do problema de se dizer se existe um homomorfismo de um grafo $G$ em um grafo $H$ é NP-completo (citar fonte?).

Para $f : G \to H$ ser um isomorfismo, $f$ deve ser uma bijeção. Assim, olhamos agora apenas para as permutações em $n$ elementos para gerar as funções canditatas em *verify_iso*. O problema se mantém com tempo de execução exponencial para o pior caso. Entretanto, ainda não se foi possível mostrar que o problema é NP-completo, sendo um grande canditato a membro da classe de problemas NP que não se encontram em P nem em NP-completo [1]. Uma implementação otimizada para esse problema, assim como para encontrar o grupo de automorfismos de um grafo, é o progrma *nauty*, de Brendan McKay, que consegue resolvê-lo para grafos com grande número de vértices [6].

```
# Listando homomorfismos de G em H
def list_homo(G, H):
    lista_homo = []
    for f in gen_func(len(G), len(H)):
        homo = Truee modo exaustivo
        # vejamos se f é homomorfismo:
        for (i, j) in edges(G):
            if H[f[i]][f[j]] == 0:
                homo = False
                break
        if homo: lista_homo.append(f)
    return lista_homo
```

```
# Listando todos os automorfismos de G
def list_aut(G):
    lista_auts = []
    for f in list(itertools.permutations(list(range(len(G))))
        :
        iso = True
        for (i, j) in edges(G):
            if G[f[i]][f[j]] == 0:
                iso = False
                break
        if iso:
            g = invert(f)
            for (i, j) in edges(G):
                if G[g[i]][g[j]] == 0:
                    iso = False
                    break
        if iso: lista_auts.append(f)
    return lista_auts
```

A partir da listagem do monóide de endomorfismos e do grupo de automorfimos de

um grafo utilizando o método de força bruta citado acima, podemos agora verificar de modo exaustivo se um grafo é um core, e encontrar seu core caso não o seja:

```
1 # verifica se G é um core comparando grupo de automorfismos
     com monoide de endomorfismos
2 def is_core(G):
3     if list_aut(G)==list_homo(G,G) :return True
4     return False
```

```
1 # encontra core de G
2 def find_core(G):
3     if is_core(G): return G
4     for i in range(len(G)):
5         H=np.delete(np.delete(G,i,0),i,1)
6         if verify_homo(G,H): return find_core(H)
```

No próximo capítulo veremos que o problema de se decidir se um grafo é um core está em NP-completo, e verificaremos casos particulares onde seja possível fazer essa verificação em tempo polinomial.

# 2. Cores

Neste capítulo apresentamos alguns resultados de [3] sobre cores, em particular um algortimo que mostra como verificar se um grafo com número de independência igual a 2 é um core em tempo polinomial.

## 2.1. Emparelhamentos

Um **emparelhamento** $M$ em um grafo $G$ é um conjunto de arestas de $G$ tal que duas arestas em $M$ não são incidentes a um mesmo vértice. Dizemos que $M$ **cobre** ou **satura** um conjunto de vértices $X \subseteq V(G)$ se cada vértice de $X$ é extremo de uma aresta de $M$. O tamanho de um emparelhamento máximo em $G$ é denotado $\nu(G)$, e um emparelhamento é dito **perfeito** se cobre $V(G)$. Todo emparelhamento perfeito é máximo.

Note que uma aresta cobre sempre exatamente dois vértices em um emparelhamento. Logo um grafo ímpar, isto é, com número ímpar de vértices, não pode possuir um emparelhamento perfeito. Do mesmo modo, para verificar se um emparelhamento $M$ é perfeito em um grafo de tamanho $n$ basta verificar se $|M| = \frac{n}{2}$.

O estudo de emparelhamentos e propriedades associadas, assim como o de algoritmos para encontrar emparelhamentos máximos, é uma área vasta dentro de teoria dos grafos, sendo conhecidos bons algoritmos que encontram emparelhamento máximo em tempo polinomial para grafos quaisquer. Mais informações sobre o tema podem ser encontradas em [5, Lovász e Plummer, 1986]. A seguir exibimos um algoritmo que encontra um emparelhamento máximo em um grafo bipartido em tempo polinomial:

```
1  # Adaptação algoritmo de fluxo máximo para encontrar
       emparelhamento máximo em um grafo bipartido.
2  def matching_bip(G):
3      n=len(G)
4      # primeiro encontrar bipartição:
5      try:
6          A,B = is_bipartite(G)
7      except:
8          print("G não é bipartido")
9          return
10     # orienta arestas de A para B:
11     for u in B:
```

```
12          G[u]*=(-1)
13      # adiciona fonte e sorvedouro:
14      fonte=np.array([0]*n)
15      for u in A:
16          fonte[u]=1
17      G=np.concatenate((G, [fonte]))
18      G=np.concatenate((G, np.transpose([(-1) * np.concatenate([
            fonte, [0]])])),1)
19      sorv = np.array([0]*(n+1))
20      for u in B:
21          sorv[u] = -1
22      G = np.concatenate((G, [sorv]))
23      G = np.concatenate((G, np.transpose([(-1) * np.concatenate
            ([sorv, [0]])])),1)
24      # encontra caminho p da fonte para sorvedouro, atualiza M
            por meio de diferença simétrica com p e altera direção
            do caminho p em G:
25      M=[]
26      p = path(G, n, n + 1)
27      while p:
28          for i in range(1,len(p)):     #altera sentido de p em G
29              G[p[i - 1]][p[i]] *= -1
30              G[p[i]][p[i - 1]] *= -1
31          p.pop(0)
32          p.pop(len(p)-1)
33          arestas_p=[]
34          for i in range(1,len(p)):
35              arestas_p.append({p[i-1],p[i]})
36          for e in arestas_p:      #dif simétrica M e arestas_p
37              if e in M: M.remove(e)
38              else: M.append(e)
39          p = path(G, n, n + 1)
40      return M
```

Um conceito um pouco mais fraco que o de emparelhamento é

## 2.2. Grafos com $\alpha(G) = 2$

Cores são importantes pois descrevem o comportamento de grafos com relação à homomorfismos. Parâmetros que podem ser caracterizados pela existência de homomorfimos de ou para um core, por exemplo, serão compartilhados entre $G$ e seu core. O teorema a seguir, devido a [3, Hell e Nešetřil, 1992], mostra que verificar se um

grafo é um core não é uma tarefa simples. Não exibiremos aqui sua demonstração por ser um tanto longa e não se encaixar no escopo do trabalho:

**Teorema 4.** *O problema de se decidir se um grafo $G$ é um core é NP-completo.*

Quando $G$ possui número de independência igual a dois, no entanto, é possível fazer essa verificação em tempo polinomial:

**Teorema 5.** *Seja $G$ um grafo não trivial com $\alpha(G) = 2$. Então são equivalentes:*

1. $G$ é um core;
2. para cada conjunto $K \subseteq V(G)$ que induz um subgrafo completo, existem mais do que $|K|$ vértices em $G$ que não são adjacentes a todo $K$;
3. o complemento de $G$ é 2-bicrítico.

# A. Classes de Complexidade

# Referências Bibliográficas


[1] Godsil, C. Royle, G. *"Algebraic Graph Theory"*, Springer, 2001.

[2] J. A. Bondy e U. S. R. Murty. *"Graph theory"*, Graduate Texts in Mathematics 244, Springer, 2001.

[3] Hell, P. Nešetřil, J. *"The core of a graph"*, Discrete Mathematics 109, 117-126, North-Holland, 1992.

[4] Hell, P. Nešetřil, J. *"Graphs and homomorphisms"*, Oxford Lecture Series in Mathematics and its Applications 28, 2004.

[5] Lovász, L. Plummer, M. D. *"Matching Theory"*, North-Holland Mathematics Studies 121, 1986.

[6] McKay, B. D. *"nauty User's Guide (Version 2.2)"*, Computer Science Department Australian National University.