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 com k vértices. 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** ϕ 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, ϕ 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 ϕ 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 ϕ 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 π 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 π de $V(G)$, definimos um grafo G/π tomando as classes de π 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/π com kernel π. Note que G/π será um grafo simples se e somente se π 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 ϕ 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 π de G, o homomorfismo natural f_π 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 anterior 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** (ou **núcleo**) **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 núcleos como os grafos que possuem o monóide de endomorfismos igual ao seu grupo de automorfismos. O núcleo de G é uma retração minimal de G, com respeito a inclusão. O exemplo mais imediato de família de núcleos 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. *$G \to H$ se e somente se $G^\bullet \to H^\bullet$.*

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 núcleos são isomorfos.*

Demonstração. Segue diretamente dos lemas 3 e 2. □

1.4. Algoritmos

Nesta seção exibiremos os algoritmos que foram desenvolvidos ao longo do trabalho acompanhando o estudo dos temas do primeiro capítulo 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.

São apresentados algoritmos para: o problema de se encontrar um homomorfismo de um grafo G para H, o problema de se dizer se dois grafos são isomorfos e o de se verificar se um grafo G é um core. Os dois primeiros são problemas em NP: dados G, H e uma função $f : G \to H$, certificar que f é um homomorfismo (ou isomorfimo) leva tempo polinomial no número de arestas de G (ou de G e H). Já o último é um problema em co-NP, pois certificamos que G não é um core por meio um homomorfismo para um subgrafo induzido.

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].

```
1  # Listando homomorfismos de G em H
2  def list_homo(G, H):
3      lista_homo = []
4      for f in gen_func(len(G), len(H)):
5          homo = Truee modo exaustivo
6          # vejamos se f é homomorfismo:
7          for (i, j) in edges(G):
8              if H[f[i]][f[j]] == 0:
9                  homo = False
10                 break
11         if homo: lista_homo.append(f)
12     return lista_homo
```

```
1  # Listando todos os automorfismos de G
2  def list_aut(G):
3      lista_auts = []
4      for f in list(itertools.permutations(list(range(len(G))))
           :
5          iso = True
6          for (i, j) in edges(G):
7              if G[f[i]][f[j]] == 0:
8                  iso = False
9                  break
10         if iso:
11             g = invert(f)
12             for (i, j) in edges(G):
```

```
13                  if G[g[i]][g[j]] == 0:
14                      iso = False
15                      break
16          if iso: lista_auts.append(f)
17      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 co-NP-completo, e verificaremos casos particulares onde seja possível fazer essa verificação em tempo polinomial.

2. Núcleos

Neste capítulo apresentamos alguns resultados de [3] sobre núcleos, em particular um algortimo que mostra como verificar se um grafo com número de independência igual a 2 é um núcleo 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. Se $v \in V(G)$ não é coberto por M, dizemos que v é **livre** (em 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, um emparelhamento M em um grafo de tamanho n é perfeito se e somente se $|M| = \frac{n}{2}$. Os teoremas abaixo exibem algumas caracterizações clássicas sobre emparelhamentos.

Definição 5. Dado G e um emparelhamento M em G, um **caminho M-alternante** em G é um caminho cujas arestas alternam entre arestas de M e de $E(G)\backslash M$. Um caminho alternante que possui os extremos livres em M é dito um **caminho M-aumentador**.

Teorema 4. *(Berge, 1957) Seja G um grafo e M um emparelhamento em G. M é máximo se e somente se G não possui um caminho M-aumentador.*

Demonstração. Seja M emparelhamento em G e $P = (1, 2, \ldots, k)$ um caminho M-aumentador. Podemos então construir um emparelhamento M' em G por meio da diferença simétrica entre P e M. Deste modo, passamos a cobrir todo P em M', tendo um emparelhamento de tamanho $|M'| = |M| + 1$.

Por outro lado, seja M é um emparelhamento não máximo em G e M^* emparelhamento máximo. Considere o subgrafo $S = G[M \triangle M^*]$. Cada vértice em G é incidente a no máximo uma aresta em M e uma aresta em M^*, e logo possui grau máximo 2 em S. Portanto S é formado apenas por caminhos e ciclos (pares), com arestas alternantes entre M e M^*. Como $|M^*| > |M|$, deve existir um caminho P em S com uma aresta a mais de M^* do que de M. Este caminho deve começar e terminar com uma aresta de M^*, e portanto começa e termina com um vértice livre em M. Logo, P é um caminho M-aumentador. □

Teorema 5. *(Hall, 1935) Seja G um grafo (A,B)-bipartido. G possui um emparelhamento que cobre A se e somente se $|N(S)| \geq |S|$ para todo $S \subseteq A$.*

Demonstração. Se existe um emparelhamento que cobre A, para cada $S \subseteq A$ existem $|S|$ vértices emparelhados aos vértices de S, e eles estão em $N(S)$. Então $|N(S)| \geq |N|$.

Por outro lado, suponha que não exista um emparelhamento que cubra todos os vértices de A. Seja M^* um emparelhamento máximo em G e $u \in A$ um vértice livre em M^*. Considere Z o conjunto de vértices alcançáveis de u por meio de caminhos M^*-alternantes. Como M^* é máximo, temos pelo teorema de Berge que u é o único vértice de Z que não é coberto por M^*. Defina $S = A \cap Z$ e $R = B \cap Z$. Note que os caminhos alternantes maximais que levam u para um vértice em Z possuem o outro extremo apenas em S. Portanto $|R| = |S| - 1$ e $R \subseteq N(S)$. No entanto, temos de fato que $R = N(S)$, pois todo vértice em $N(S)$ possui um caminho M^*-alternante até u. Assim, concluímos que $|N(S)| = |R| = |S| - 1$, e a condição do teorema de Hall falha. □

Corolário 3. *Um grafo (A,B)-bipartido possui um emparelhamento perfeito se e somente se $|A| = |B|$ e $|N(S)| \geq |S|$ para todo $S \subseteq A$.*

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. Informações detalhadas sobre o tema podem ser encontradas em Lovász e Plummer [5]. A seguir exibimos um algoritmo que encontra um emparelhamento máximo em um grafo bipartido G com bipartição A, B em tempo polinomial:

1 Oriente as arestas de G de A para B, adicione uma fonte f ligada a todos os vértices de A e um sorvedouro s ligado a todos os vértices de B;
2 M = vazio;
3 Enquanto houver caminho P de f até s faça:
4 Atualize M por meio da diferença simétrica com P, ignorando as direções das arestas e as arestas ligadas a f e s;
5 Inverta o sentido do caminho P em G;
6 Retorne M.

Algorithm 1: EMP_BIPARTIDO

Note que, ao final de cada iteração do laço principal do algoritmo, qualquer caminho M-aumentador em G induz um caminho P de f até s. Portanto, ao final do algoritmo não existem caminhos M-aumentadores em G, e pelo teorema de Berge M é máximo.

Definimos agora um conceito que pode ser visto como um relaxamento do conceito de emparelhamento: um **2-emparelhamento** W em G é uma atribuição de

pesos 0, 1 ou 2 às arestas de G tal que a soma de pesos de todas as arestas incidentes em um vértice é no máximo 2. Dizemos que as arestas de peso 1 e 2 "estão" no 2-emparelhamento W, enquanto as de peso 0 não. O **tamanho** de um 2-emparelhamento W é a soma de todos os pesos das arestas e denotado $|W|$. O tamanho de um 2-emparelhamento máximo em G é denotado $\nu_2(G)$. Note que qualquer emparelhamento pode ser visto como um 2-emparelhamento se atribuirmos valor 2 às suas arestas. No entanto, o conceito de 2-emparelhamento é mais geral, já que podemos cobrir por exemplo um ciclo ímpar com ele. Um 2-emparelhamento em um grafo G é dito **básico** se as suas arestas de peso 1 formam apenas ciclos ímpares.

Dizemos que um 2-emparelhamento é **perfeito** se a soma de pesos das arestas em cada vértice é exatamente 2. Um 2-emparelhamento é perfeito se e somente se possui tamanho igual $|V(G)|$.

Lema 4. *Todo grafo possui um 2-emparelhamento máximo que é básico.*

Demonstração. Seja W um 2-emparelhamento em G. Se $P_k = (1, 2, \dots k)$ é um caminho de tamanho k em W tal que suas arestas possuem todas peso 1, podemos construir um novo 2-emparelhamento de tamanho maior ou igual W dobrando o peso das arestas $12, 34, \dots$ e reduzindo o peso das arestas restantes em P_k pra zero. Do mesmo modo, se possuímos um ciclo par com pesos 1 em W, obtemos um 2-emparelhamento de mesmo peso dobrando e zerando as arestas do ciclo alternadamente. □

Podemos computar um 2-emparelhamento máximo em um grafo G qualquer tão eficientemente quanto computamos um emparelhamento máximo em um grafo bipartido. Para isso, primeiro vamos construir um grafo bipartido G^b a partir de G da seguinte forma: Para cada vértice v de G, defina vértices v' e v'' em G^b, e para cada aresta uv de G construímos as arestas $u'v''$ e $v'u''$ em G^b. O teorema a seguir nos mostra uma correspondência entre 2-emparelhamentos máximos em G e emparelhamentos máximos em G^b:

Lema 5. *Seja G um grafo e G^b construído como acima. Então $\nu_2(G) = \nu(G^b)$.*

Demonstração. Seja M é um emparelhamento máximo em G^b. Defina o peso de uv em G como o valor $|M \cap \{u'v'', v'u''\}|$. A soma dos pesos em um vértice u será no máximo 2, pois u' e u'' são cobertos por no máximo uma aresta em M. Temos então um 2-emparelhamento em G de mesmo tamanho que M. Logo, $\nu_2(G) \geq \nu(G^b)$.

Por outro lado, seja W um 2-emparelhamento máximo em G. Pelo 4 podemos assumir que W é básico. Vamos construir um emparelhamento M em G^b. Se $W(uv) = 2$, adicione as arestas $u'v''$ e $v'u''$ a M. Se $C_n = (1, 2, \dots, n, 1)$ é um ciclo ímpar de pesos 1 em W, adicione as arestas $1'2'', 2'3'', \dots, n'1''$ a M. M construído desta forma é um emparelhamento de G^b com mesmo tamanho de W, e portanto $\nu(G^b) \geq \nu_2(G)$. □

Definição 6. Um grafo G é **2-bicrítico** se $|V(G)| \geq 2$ e para qualquer $v \in V(G)$ o grafo $G - v$ contém um 2-emparelhamento perfeito.

O teorema a seguir nos dá um análogo do corolário 3 do teorema de Hall para 2-emparelhamentos, e consequentemente uma caracterização também para grafos 2-bicríticos:

Teorema 6. *[5] Um grafo G possui um 2-emparelhamento perfeito se e somente se* $|N(A)| \geq |A|$ *para todo conjunto independente A em G.*

Corolário 4. *Um grafo G é 2-bicrítico se e somente se* $|N(A)| > |A|$ *para todo conjunto independente não vazio A em G.*

Demonstração. Segue diretamente da definição de 2-bicrítico e do teorema anterior. □

2.2. Complexidade

Núcleos são importantes pois descrevem o comportamento de grafos com relação a homomorfismos. Parâmetros que podem ser caracterizados pela existência de homomorfimos de ou para um núcleo, por exemplo, serão compartilhados entre G e seu núcleo. O próximo teorema, devido a Hell e Nešetřil [3], mostra que verificar se um grafo é um núcleo não é uma tarefa simples. Não exibiremos aqui sua demonstração completa por ser um tanto longa, apenas um esboço da ideia. Para um grafo arbitrário H fixo, definimos o problema:

Problema. $CORE_H$: Dado um grafo G e um homomorfismo $G \to H$, decidir se existe um homomorfismo não sobrejetivo $G \mapsto G$.

Esta formulação tem a vantagem de nos dar um problema em NP, dado que é fácil verificar se um homomorfismo $G \mapsto G$ é não sobrejetivo. Além disso, núcleos são os grafos para os quais a resposta para o problema é NÃO. Note que esta formulação nos dá na realidade uma família de problemas de decisão, pois depende da escolha de uma grafo H a princípio.

Teorema 7. *Se H é bipartido, o problema* $CORE_H$ *está em P. Caso contrário, está em NP-completo.*

Demonstração. (Esboço) Se H é bipartido, então $G \to H \to K_2$ é homomorfismo, e logo G também também é bipartido. Mas para um grafo bipartido, ser um núcleo é o mesmo que ser isomorfo a K_2, logo facilmente verificável.

Se H não é bipartido, considere g a cintura ímpar de H. Sejam C_g e C_{g+2} os ciclos de tamanho g e $g+2$ respectivamente. É conhecido que reconhecer grafos que possuem um homomorfismo para C_{g+2} (e para qualquer grafo não bipartido) é NP-completo [4]. A prova segue então em dois passos:

1. Mostre que reconhecer se um grafo possui um homomorfismo para C_{g+2} ainda é NP-completo dentre os grafos que possuam cintura pelo menos $g+6$ e que admitam homomorfismo para C_g.

2. Considere G um grafo com cintura maior ou igual $g+6$ que admite homomorfismo para C_g. Construa um grafo G^* que admite homomorfimo para H, e que não é um core se e somente se G admite um homomorfismo para C_{g+2}.

O modo como G^* é construído garante que essa é uma redução polinomial. □

Note que um grafo G pode ser entrada para $CORE_H$ somente se existe homomorfismo $G \to H$, ou seja, G é H-colorível. Assim, o teorema acima nos mostra a dicotomia do problema de se verificar se H é um núcleo com relação a H-colorabilidade: se G é H-colorível para H bipartido (equivalente a ser K_2 colorível), o problema está em P; caso contrário, está em co-NP-completo.

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

Teorema 8. *Seja G um grafo não trivial com $\alpha(G) = 2$. Então são equivalentes:*

1. *G é um core;*
2. *para cada clique K de G, existem mais do que $|K|$ vértices em G que não são adjacentes a todos os elementos de K;*
3. *o complemento de G é 2-bicrítico.*

Antes de demonstrá-lo, precisamos exibir uma versão de natureza combinatória do teorema de Hall, formulada para coleções de conjunto finitos. Essa versão é na realidade a primeiro forma em que Hall apresentou seu teorema, e é equivalente à versão para grafos que apresentamos no teorema 5.

Seja A um conjunto finito e $\mathcal{A} = \{A_i : i \in I\}$ uma família finita de subconjuntos de A, não necessariamente distintos. Um **sistema de representantes distintos (SDR)** para a família $\mathcal{A}$ é um conjunto $\{a_i : i \in I\}$ de elementos distintos de A tal que $a_i \in A_i$ para todo $i \in I$. Em outras palavras, um SDR seleciona um representante distinto para cada A_i em $\mathcal{A}$. O teorema de Hall se enuncia então da seguinte forma:

Teorema 9. *(Hall, 1935) Seja $\mathcal{A} = \{A_i : i \in I\}$ uma família de subconjuntos de A como descrita acima. $\mathcal{A}$ possui um sistema de representantes distintos se e somente se para cada subconjunto $J \subseteq I$ vale:*

$$\Big|\bigcup_{i \in J} A_i\Big| \geq |J|.$$

Demonstração. Para ver que esta formulação é equivalente ao teorema 5, defina um grafo G com bipartição X, Y onde $X := I$, $Y := A$ e para cada vértice i em X definimos $N(i) := A_i$. Um SDR de $\mathcal{A}$ é deste modo um emparelhamento que cobre X em G. □

A condição enunciada no teorema de Hall para a família $\mathcal{A}$ é chamada **condição de casamento**.

Demonstração. (teorema 8) 2 $\implies$ 1: Assuma que G não é um core, e seja $f : G \to G'$ uma retração de G em um subgrafo próprio. Para cada $x \in G'$, $f^{-1}(x)$ é um conjunto independente, e portanto possui no máximo 2 elementos. Defina K como o conjunto de vértices $x \in G'$ tais que $f^{-1}(x)$ possui exatamente 2 elementos; isto é, x é a imagem de mais um elemento além de si próprio por f. K é um clique, pois se $x, y \in K$ e $x \nsim y$,

então $f^{-1}(x) \cup f^{-1}(y)$ possuiria 4 elementos. Pelo mesmo argumento, cada vértice de G' está ligado a todo K, caso contrário existiria conjunto independente com pelo menos 3 elementos em G. Portanto, os únicos vértices de G que não estão ligados a todo K estão em $G - G'$, que possui exatamente $|K|$ elementos.

$1 \implies 2$:Suponha que 2 é falso, i.e., existe clique K tal que o número de vértices que não é adjacente a todo K é $\leq |K|$. Podemos assumir K minimal: cada subconjunto próprio K' de K possui mais do que $|K'|$ vértices não adjacentes a todo K'. Então K possui exatamente $|K|$ vértices não adjacentes a todo K.

Para cada $x \in K$, considere o conjunto $S_x = \{y \in V(G) : y \text{ não é adjacente a } x\}$. Note que o conjunto $\bigcup_{x \in K'} S_x$ simboliza o conjunto de vértices não adjacentes a todo K'. Pelo que construímos no parágrafo anterior, a família $S = \{S_x : x \in K\}$ satisfaz a condição de casamento do teorema de Hall, pois para todo $K' \subseteq K$ vale:

$$\Big|\bigcup_{x \in K'} S_x\Big| \geq |K'|.$$

Portanto, pelo teorema de Hall, podemos tomar representantes distintos para cada S_x, digamos y_x.

Vejamos que a aplicação f que leva cada y_x em x e mantém todos os outros vértices fixos é uma retração: precisamos verificar apenas que as adjacências relacionadas aos vértices y_x são mantidas. Seja $z \sim y_x$. Se $z = y_w$ para algum w, então x e w estão em K e $x \sim w$. Caso contrário, z está ligado a todo K, e $f(z) = z \sim x$. Portanto f é uma retração e G naõ é um core.

$2 \iff 3$: A equivalência entre os itens 2 e 3 é dada no corolário 4, apenas enunciado aqui para o grafo complementar. □

2.3. Algoritmos

```
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
```

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.