Uma relação entre b-coloração e teoria espectral em cografos

Átila Arueira Jones
Universidade Federal Fluminense / Instituto Federal Sudeste de Minas Gerais
Niterói - RJ / Juiz de Fora - MG
atilajones@id.uff.br

Fábio Protti
Universidade Federal Fluminense
Niterói - RJ
fabio@ic.uff.br

Renata Raposo Del-Vecchio
Universidade Federal Fluminense
Niterói - RJ
renata@vm.uff.br

RESUMO

Nesse trabalho sugerimos uma nova cota para o número b-cromático de um cografo conexo, dada em termos do índice do grafo (maior autovalor da matriz de adjacência). Para estabelecer a conjectura, usamos um algoritmo gerador de cografos para verificar que a cota funciona.

PALAVRAS CHAVE. Cografos, b-coloração, Teoria Espectral de Grafos.

Área principal. Teoria e Algoritmos em Grafos (TAG)

ABSTRACT

In this work we suggest a new upper bound for the b-chromatic number of a connected cograph, is given in terms of the index of the graph (the largest eigenvalue of its adjacency matrix). To provide a conjecture, we use the cograph generation algorihm to verify that this bound works.

KEYWORDS. Cographs, b-coloring, Spectral Graph Theory.

Main area. Theory and Algorithms in Graphs (TAG)

1. Introdução

Uma variação do problema de coloração é a **b-coloração** de um grafo, que consiste numa coloração própria onde cada cor deve possuir ao menos um vértice dominante associado, isto é, adjacente a pelos um vértice de cada uma das outras cores. O número b-cromático foi introduzido em Irving e Manlove [1999] definido como o maior número de cores tal que o grafo admite uma b-coloração, e é denotado por $\chi_b(G)$. Foi provado que determinar $\chi_b(G)$ é NP-difícil para um grafo qualquer, então é natural a busca de cotas para o número b-cromático de um grafo, como em Alkhateeb e Kohl [2011] e Kouider e Mahéo [2002].

Encontrar cotas para o número cromático envolvendo outros parâmetros já desperta interesse há mais de 40 anos, inclusive envolvendo o índice do grafo. A saber, dado um grafo $G(V, E)$ podemos associá-lo a uma matriz; neste caso trabalharemos com a matriz de adjacência definida como uma matriz de ordem $|V|$ onde cada entrada vale 1 caso os vértices correspondentes sejam adjacentes, e 0 caso contrário. A matriz de adjacência é simétrica e, portanto, sempre admite autovalores reais, que por sua vez são definidos como autovalores do grafo G; o maior deles é chamado índice do grafo e denotado por $\lambda(G)$, também conhecido como raio espectral. A teoria espectral de grafos consiste no estudo de autovalores de matrizes associadas a grafos.

Algumas cotas para o número cromático envolvendo o índice de um grafo podem ser encontradas em Cvetkovic [1972]. Quanto ao número b-cromático, não se vê na literatura nenhuma cota relacionada com tal parâmetro, o que define o objetivo deste trabalho.

Neste trabalho propomos uma desigualdade entre índice do grafo e o número b-cromático para cografos conexos. A saber, os cografos são conhecidos por grafos livres de P_4. Para verificar a validade da relação proposta, usamos o algoritmo construído em Jones et al. [2016], capaz de gerar todos os cografos não isomorfos dois a dois com n vértices, cujo atraso é $O(n)$.

Inicialmente faremos uma breve introdução com os resultados necessários para compreensão da desigualdade estimada. Em seguida descreveremos os algoritmos usados na implementação, detalharemos a execução dos testes aplicados e os resultados obtidos.

2. Resultados preliminares

Um dos temas clássicos da teoria de grafos é o problema de coloração de vértices e suas variações. Formalmente uma **k-coloração** de um grafo $G(V, E)$ nada mais é que uma função sobrejetiva $c : V(G) \to \{1, \ldots, k\}$ e a **classe da cor** i é o conjunto $C_i = \{v \in V(G); c(v) = i\}$, onde os valores $1, \ldots, k$ são chamados cores. A restrição mais natural a ser imposta é usar k cores de forma que vértices adjacentes tenham cores distintas, chamada de **k-coloração própria**, o que é resolvido facilmente atribuindo uma cor para cada vértice. Mas o problema de fato é obter a solução ótima, ou seja, o menor número de cores com essa propriedade, chamado **número cromático** do grafo, denotado por $\chi(G)$. Irving e Manlove (Irving e Manlove [1999]) introduziram uma variação do problema: a b-coloração.

Dado um grafo G colorido com k cores, dizemos que um vértice é **b-dominante** se é adjacente a pelo menos um vértice de cada uma das outras $k - 1$ cores. Uma **b-coloração** é uma coloração onde cada classe de cor possui ao menos um vértice b-dominante. Formalmente:

Definição 2.1. *Uma **b-coloração** é uma k-coloração tal que $\forall i \in \{1, \ldots, k\}$, $\exists u \in C_i$ (b-dominante) tal que $\forall j \in \{1, \ldots, k\} \backslash \{i\}$, $\exists v \in C_j$ com $uv \in E(G)$. O número **b-cromático**, denotado por $\chi_b(G)$, é o maior k onde o grafo admite uma b-coloração com k cores.*

Exemplo 2.1. *Na Figura 1 temos o grafo G à esquerda com uma 3-coloração e à direita G b-colorido com 4 cores (b-dominantes em destaque). Neste exemplo é fácil ver que $\chi(G) = 3$ e $\chi_b(G) = 4$.*

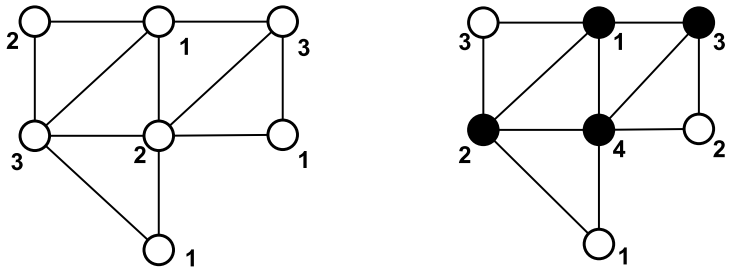

Figura 1: Exemplo número cromático e b-cromático

Irving e Manlove introduzem também o conceito de **m-grau** de um grafo G, denotado por $m(G)$, como o maior inteiro m tal que o grafo possua m vértices de grau maior ou igual a $m-1$. Isto é, ao ordernar os graus dos vértices em $d(v_1) \geq \ldots \geq d(v_n)$ definimos $m(G) = max\{i; d(v_i) \geq i-1,\ 1 \leq i \leq n\}$. No exemplo anterior temos $m(G) = 4$.

Proposição 2.1. *Irving e Manlove [1999] Para todo grafo G vale* $\chi(G) \leq \chi_b(G) \leq m(G)$.

Demonstração. De fato se $\chi_b(G) = k$, então então existem pelo menos k vértices b-dominantes em G, onde cada um tem grau pelo menos $k-1$, donde segue a segunda desigualdade. Para a primeira basta observar que $\chi(G)$ é o menor número de cores tal que grafo admite coloração própria. □

A classe de grafos que será tratada neste trabalho é a dos cografos, descoberta independentemente por diversos autores desde 1970. Os cografos são usualmente definidos como grafos livres de P_4, segundo equivalência provada em [Corneil et al., 1981]. Mas sua definição original é apresentada em forma recursiva: o grafo trivial é cografo, o complementar de cografo é cografo, a união de cografos é cografo.

Vale lembrar que o **join** entre dois grafos $G_1(V_1, E_1)$ e $G_2(V_2, E_2)$ é o grafo $G_1 \vee G_2$ cujo conjunto de vértices é $V_1 \cup V_2$ e arestas $E_1 \cup E_2 \cup \{xy; x \in E_1 \text{ e } y \in E_2\}$; tal operação também é chamada de **união complementar**, pois vale $\overline{G_1 \cup G_2} = \overline{G_1} \vee \overline{G_2}$. Então um cografo pode ser obtido por sucessivas operações de *join* e *união* entre cografos. Tal recurso foi utilizado em [Corneil et al., 1984] para introduzir a representação de cografo através de uma árvore, chamada **coárvore**, cujas folhas são os vértices do grafo e os vértices internos possuem ao menos dois filhos e representam operações de *join* ou *união*, rotulados por $tipo$-1 e $tipo$-0, respectivamente. Além disso a coárvore deve ser escrita de forma que os vértices que compõem um caminho possuem seus tipos alternados, o que garante que cada cografo tenha somente uma coárvore associada, a menos de permutação entre vértices; por outro lado temos que cada coárvore é referente a um único cografo.

Como os cografos são identificados por árvores, vamos fixar algumas notações acerca de uma árvore enraizada T: a raiz será denotada por $raiz(T)$; dado um vértice v (diferente da raiz) denotamos P_v o único caminho de v até a raiz; os vértices de P_v (exceto v) são ditos **ancestrais** de v em T, onde o antecessor imediato de v em P_v é chamado **pai** de v e denotado por $pai_T(v)$; fixamos $pai_T(raiz(T)) =$ ***null*** ; para cada vértice interno v definimos seus **filhos** pelo conjunto $filhos_T(v) = \{u \in V(T); pai_T(u) = v\}$ e a **irmandade de** v por $I_T(v) = \{filhos_T(pai_T(v))\}$, para $v \neq raiz(T)$, fixando $I_T(raiz(T)) = \{raiz(T)\}$; denotaremos por $T(v)$ a subárvore induzida por T cuja raiz é v. Em casos livres de ambiguidade ocultaremos o índice T das notações acima. Além disso denotaremos o isomorfismo entre dois grafos G_1 e G_2 por $G_1 \equiv G_2$.

3. A desigualdade proposta

O estudo de cotas envolvendo o número cromático de um grafo e o seu índice desperta interesse há mais de 40 anos, como pode ser visto em Cvetković e Rowlinson [1990]. Em particular Wilf provou o seguinte teorema:

Teorema 3.1. *Wilf [1967] Todo grafo G satisfaz* $\chi(G) \leq \lambda(G)+1$, *onde* $\lambda(G)$ *é o maior autovalor da matriz de adjacência de G.*

Em relação ao número b-cromático também já existem resultados envolvendo cotas, como em Alkhateeb e Kohl [2011], mas ainda não há estudos sobre cotas envolvendo os autovalores de um grafo.

Para grafos em geral determinar o parâmetro $\chi_b(G)$ é NP-difícil, como provado por Irving e Manlove [1999]. Contudo os cografos apresentam bom comportamento diante deste problema, uma vez que $\chi_b(G)$ pode ser obtido em tempo polinomial para um cografo G, provado em Bonomo et al. [2009]. Motivado por essa estreita relação e a desigualdade apresentada pelo teorema anterior levantamos a seguinte questão:

$$\textit{Todo cografo conexo } G \textit{ satisfaz } \chi_b(G) \leq \lambda(G) + 1. \qquad (*_1)$$

A questão acima sugere uma desigualdade razoável, já que a igualdade é atingida para grafos completos, pois são cografos e satisfazem $\lambda(K_n) = n - 1$ e $\chi_b(K_n) = n$. Em contrapartida, observe que para 4 cópias da estrela S_3, temos $\lambda(4S_3) = \sqrt{3}$ e $\chi_b(4S_3) = 4$, donde $\chi_b > \lambda + 1$, o que justifica a inclusão da hipótese de conexidade sobre a questão envolvida.

4. Algoritmos implementados

A fim de verificar a existência de contra-exemplos para a afirmação $(*_1)$, efetuamos testes computacionais fazendo uma busca nos cografos conexos, fazendo uso do algoritmo gerador de cografos desenvolvido em Jones et al. [2016]. Nas subseções seguintes detalharemos a estrutura de dados utilizada, bem como descreveremos os algoritmos implementados.

4.1. Estrutura de dados do cografo

Como todo cografo é representado por uma árvore e os parâmetros que desejamos podem ser obtidos diretamente da árvore, é natural representarmos o cografo através da mesma, que por sua vez será implementada através de listas encadeadas de vértices, cuja estrutura básica é descrita abaixo.

Algorithm 4.1 Estrutura do Vértice

```
1:  procedure Vertice
2:      Vertice Pai;
3:      Vetor < Vertice > Filhos;
4:      double Diag;
5:      integer FolhasInduzidas;
6:      integer label;                          ▷ Nó interno: 0, 1. Folha: 2
7:      integer Cromatico;
8:      integer Nivel;
9:      integer Grau;
10: end function
```

Dada uma árvore T e um vértice estruturado como acima, detalhamos abaixo a função de cada campo.

- vértice *Pai* armazenará $pai(v)$;
- Vetor *Filhos* armazenará $Filhos(v)$;
- escalar *Diag* será visto posteriormente, cuja utilidade é obtenção de estimativa para o $\lambda(G)$;
- inteiro *FolhasInduzidas* armazena o número de folhas induzidas por $T(v)$;
- inteiro *label* armazena se o vértice é interno ($tipo$-1 ou $tipo$-0) ou uma folha (denotado por $tipo - 2$)
- inteiro *Cromatico* armazena o número cromático do cografo cuja coárvore é $T(v)$, será visto em detalhes na seção seguinte.

- inteiro *Nivel* armazena o valor $nivel(v)$;
- inteiro *Grau* armazena o grau do vértice v no cografo referente a T.

A árvore é acessada através de seu vértice raiz, que permite operar e obter informações de toda a árvore.

O acesso a qualquer campo da estrutura vértice será denotado por ".". Por exemplo ao acessar o *label* de um vértice v, usaremos a notação $v.label$.

O preenchimento do campo *FolhasInduzidas* pode ser feito de forma recursiva com procedimento iniciado nas folhas da árvore. O campo *nivel* é a aplicação direta da definição, ou seja, a raiz tem nível 1 e a cada novo vértice v instanciado, atribuímos $v.nivel \leftarrow v.pai.nivel + 1$.

A seguir detalhamos o preenchimento do campo *Grau*. Observe que não nos referimos ao grau do vértice na árvore, mas sim no vértice correspondente no cografo associado a esta árvore; portanto este campo só faz sentido nas folhas da árvore, para os demais vértices o campo é útil no procedimento.

Algorithm 4.2 Calcula o grau (cografo) de cada vértice
Input: Vértice v de uma coárvore T
Output: Calcula o grau de cada vértice na subárvore $T(v)$

```
1:  procedure CalculaGrau(v)
2:      if v.label = 1 then
3:          for all w ∈ v.Filhos do
4:              aux ← Σ_{x∈I(w)\{w}} x.FolhasInduzidas
5:              w.Grau ← v.Grau + aux
6:              CalculaGrau(w)
7:          end for
8:      else
9:          if v.label = 0 then
10:             for all w ∈ v.Filhos do
11:                 w.Grau ← v.Grau
12:                 CalculaGrau(w)
13:             end for
14:         end if
15:     end if
16: end function
```

Omitiremos mais detalhes sobre o preenchimento dos demais campos, pois fogem do objetivo do presente texto. Daqui em diante assumiremos que todo vértice já possui seus campos *FolhasInduzidas*, *Nivel*, *Pai*, *Filhos*, *label* e *Grau* devidamente preenchidos.

4.2. Cálculo do número b-cromático

Em Bonomo et al. [2009] é provado que o cálculo de $\chi_b(G)$ é polinomial para cografos. Usaremos a estratégia proposta no artigo para obter o número b-cromático de forma eficiente. Antes de descrevermos o procedimento, é necessário introduzir o conceito de vetor b-dominante.

Definição 4.1. *Bonomo et al. [2009] Dado um grafo G, o vetor b-dominante dom_G é um vetor onde para cada t, $\chi(G) \leq t \leq |V(G)|$, a entrada $dom_G[t]$ é o maior número de classes de cores que possuem vértice b-dominante dentre qualquer coloração própria de G com t cores. Para valores de t fora da intervalo, observe que $dom_G[t] = 0$ para $t > |V(G)|$ e $dom_G[t]$ não faz sentido para valores $t < \chi(G)$.*

Exemplo 4.1. *O vetor b-dominante do grafo abaixo é* $(3, 3, 2, 0)$*, onde* $\chi(G) = 3$.

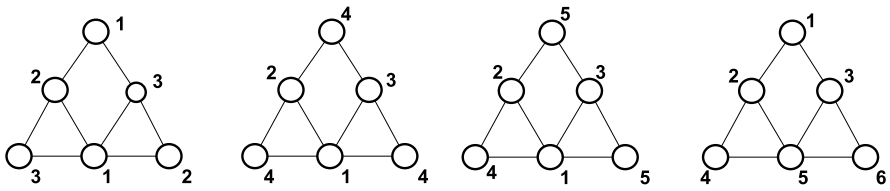

Figura 2: Grafo colorido com 3,4,5 e 6 cores, nesta ordem.

Observe que o número b-cromático de um grafo G é o maior inteiro k onde $dom_G[k] = k$. Para o cálculo do número b-cromático em cografos, nos apoiamos nos teoremas desenvolvidos no artigo, a saber:

Teorema 4.1. *Bonomo et al. [2009] Sejam* $G_1(V_1, E_1)$ *e* $G_2(V_2, E_2)$ *dois grafos tais que* $V_1 \cap V_2 = \emptyset$*. Se* $G = G_1 \cup G_2$ *e* $t \geq \chi(G)$*, então*

$$dom_G[t] = min\{t, dom_{G_1}[t] + dom_{G_2}[t]\}$$

Teorema 4.2. *Bonomo et al. [2009] Sejam* $G_1(V_1, E_1)$ *e* $G_2(V_2, E_2)$ *grafos tais que* $V_1 \cap V_2 = \emptyset$*. Sejam* $G = G_1 \vee G_2$ *e* $\chi(G) \leq t \leq |V(G)|$ *e considere ainda* $a = max\{\chi(G_1), t - |V_2|\}$ *e* $b = \{|V_1|, t - \chi(G_2)\}$*. Então* $a \leq b$ *e*

$$dom_G[t] = max\{dom_{G_1}[j] + dom_{G_2}[t - j]; a \leq j \leq b\}$$

Observando o enunciado dos Teoremas 4.1 e 4.2 vemos que é necessário calcular inicialmente o número cromático de cografos, onde nos apoiamos no teorema de Corneil *et al*:

Teorema 4.3. *Corneil et al. [1984] Se* G *é o grafo trivial, então* $\chi(G) = 1$*. Sejam* $G_1(V_1, E_1)$ *e* $G_2(V_2, E_2)$ *dois grafos tais que* $V_1 \cap V_2 = \emptyset$*. Então* $\chi(G_1 \cup G_2) = max\{\chi(G_1), \chi(G_2)\}$ *e* $\chi(G_1 \vee G_2) = \chi(G_1) + \chi(G_2)$

Os Teoremas 4.3, 4.2 e 4.1 deixam claro como calcular o número cromático e o vetor b-dominante em cada vértice da árvore, e consequentemente obter o número b-cromático de um cografo. Tal procedimento denominaremos $Bcromatico(T)$.

4.3. Cálculo do índice

A obtenção do índice de um grafo G com n vértices envolve o cálculo de determinantes de ordem n e de raízes de polinômios de grau n, o que pode ser custoso computacionalmente. Como nosso propósito é executar vários testes a fim de verificar a afirmação $(*_1)$, o cálculo pela definição não é um caminho viável. Como só queremos verificar uma desigualdade utilizamos um método mais eficiente capaz de majorar os autovalores.

Em Jacobs et al. [2017] é proposto um algoritmo de localização de autovalores em cografos, cuja entrada é uma coárvore T na forma *standard*, isto é, coárvore estruturada de forma que todas as folhas estejam no último nível. O procedimento que transforma uma coárvore numa coárvore *standard* se baseia em inserir vértices intermediários que não alteram a estrutura do cografo, até que todas as folhas sejam postas no último nível. A seguir exibimos um exemplo da aplicação deste procedimento.

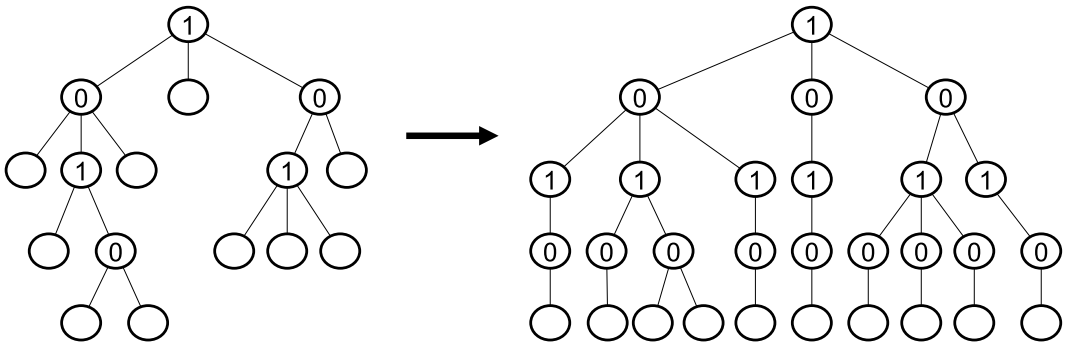

Figura 3: Coárvore T e sua versão *standard*

Exemplo 4.2.

A ideia do algoritmo proposto em Jacobs et al. [2017] é a seguinte: dada uma coárvore *standard* T e um valor real x, a cada folha v de T é associado um número $v.diag$, chamado valor diagonal, inicializado com x. Então são realizados dois tipos de operações: atualização dos valores $v.diag$ e remoção de adjacências em T, onde implicitamente tais processos são equivalentes a operações na matriz $M = A(G) + x.\mathbb{I}$, onde $\mathbb{I}$ é a matriz identidade. Após a execução, cada vértice terá seu valor diagonal definitivo, o que equivale a M estar na sua forma diagonalizada, cujos autovalores são os elmentos da diagonal, representado pelo campo $v.diag$. Então é garantido que:

- A quantidade de autovalores de $A(G)$ menores que $-x$ coincide com a quantidade de folhas com valor diagonal negativo.
- A quantidade de autovalores de $A(G)$ iguais a $-x$ coincide com a quantidade de folhas com valor diagonal nulo.
- A quantidade de autovalores de $A(G)$ maiores que $-x$ coincide com a quantidade de folhas com valor diagonal positivo.

Como o nosso objetivo é verificar a validade da afirmação $(*_1)$, ou seja, constatar se $\lambda(G)$ é maior que o parâmetro $\chi_b(G) - 1$, basta aplicar o valor $x = 1 - \chi_b(G)$ como entrada do procedimento e verificarmos se no final do processo existe algum vértice com valor diagonal positivo ou nulo. Em caso afirmativo, existe ao menos um autovalor maior ou igual a $-x$, portanto $\lambda(G) \geq -x$ confirmando a afirmação. Por outro lado, um contra exemplo para a afirmação é um cografo G que após a aplicação do procedimento com entrada x, todos os vértices possuem valor diagonal negativo, constatando que todos os autovalores, inclusive o índice, são menores que $-x$. Novamente vale ressaltar que para aplicar o algoritmo não é necessário em momento algum a estrutura do cografo, pois todas as operações são feitas sobre a coárvore.

A seguir escrevemos o procedimento proposto em Jacobs et al. [2017]. Antes observe que as n folhas da árvore estão localizadas no nível $m := nivel(T)$, as quais rotulamos em ordem decrescente de índice: $v_k, v_{k-1}, \ldots, v_2, v_1$.

Algorithm 4.3 Jacobs et al. [2017] Algoritmo diagonalização

Input: Coárvore *standard* T e real x
Output: Cálculo do valor diagonal de cada vértice

procedure $LocalizacaoAutovalores(T, x)$
  For each v folha de T **do** $v_k.Diag \leftarrow x$
  for i $\leftarrow m-1$ **to** 1 **do** ▷ análise do nível i.
    while $\exists v_k, v_l$ $(k < l)$ onde $nivel(lca(v_k, v_l)) = i$ **do**
      $\beta \leftarrow v_k.Diag$
      $\alpha \leftarrow v_l.Diag$
      if $lca(v_k, v_l)$ é $tipo - 1$ **then**
        if $\alpha + \beta \neq 2$ **then**
          $v_l.Diag \leftarrow \frac{\alpha\beta - 1}{\alpha + \beta - 2}$ $v_k.Diag \leftarrow \alpha + \beta - 2$
        else
          if $\beta = 1$ **then**
            $v_l.Diag \leftarrow 1$ $v_k.Diag = 0$
          else
            $v_l.Diag \leftarrow 1$ $v_k.Diag = -(1-\beta)^2$
            remova adjacências entre v_l e $lca(v_k, v_l)$
          end if
        end if
        remova adjacências entre v_k e $lca(v_k, v_l)$
      else
        if $\alpha + \beta \neq 0$ **then**
          $v_l.Diag \leftarrow \frac{\alpha\beta}{\alpha + \beta}$ $v_k.Diag \leftarrow \alpha + \beta$
        else
          if $\beta = 0$ **then**
            $v_l.Diag \leftarrow 0$ $v_k.Diag = 0$
          else
            $v_l.Diag \leftarrow \beta$ $v_k.Diag = -\beta$
            remova adjacências entre v_l e $lca(v_k, v_l)$
          end if
        end if
        remova adjacências entre v_k e $lca(v_k, v_l)$
      end if
    end while
  end for
end function

Mais detalhes sobre o funcionamento do algoritmo acima podem ser vistos em Jacobs et al. [2017].

Nosso interesse é verificar se o índice de um grafo é maior que um específico valor x dado como entrada, o que é feito pelo algoritmo abaixo.

Algorithm 4.4 Verifica se o índice é maior que x

Input: Coárvore T e escalar x
Output: Valor *booleano*

```
1: procedure IndiceMaiorQue(T, x)
2:     T ← Standard(T)
3:     LocalizacaoAutovalores(T, −x)
4:     for all v folha de T do
5:         se v.Diag ≥ 0 então return true
6:     end for
7:     return false
8: end function
```

4.4. Algoritmo Gerador de Cografos

Em Jones et al. [2016] é introduzida uma ordenação de vértices e coárvores para construir um algoritmo capaz de gerar todos os cografos (não isomorfos) com n vértices. Além disso a geração pode ser feita em todos os cografos ou em apenas os conexos (ou desconexos), onde a complexidade de tempo do atraso da geração é $O(n)$. Vale ressaltar que a geração é feita diretamente na coárvore do grafo, sem a necessidade da construção do cografo em si.

A construção dos cografos é feita basicamente da seguinte forma: o procedimento inicia na menor coárvore com n vértices e em seguida é gerada a árvore imediatamente maior que a antecessora, até alcançar a maior árvore do conjunto. Esse algoritmo da construção da árvore imediatamente maior será chamado $ArvoreSeguinte(T)$. Mais detalhes podem encontrados em Jones et al. [2016].

5. Verificação da afirmação $(*_1)$

Munidos dos algoritmos descritos nas seções acima podemos construir um algoritmo capaz de verificar a desigualdade $(*_1)$ em cada cografo conexo com número fixado de vértices. O algoritmo abaixo descreve como é feita tal verificação:

Algorithm 5.5 Verificação da desigualdade $(*_1)$ com geração

Output: Grafo contraexemplo

```
1: procedure ValidacaoDesigualdadeBcromatico
2:     n = 2
3:     while true do
4:         Seja T a menor coárvore com n vértices
5:         repeat
6:             se !(IndiceMaiorQue(T, Bcromatico(T) − 1)) então stop;
7:             T ← ArvoreSeguinte(T)
8:         until T = null
9:         n ← n + 1
10:    end while
11: end function
```

Todos os testes foram feitos através dos algoritmos descritos neste trabalho, implementados na linguagem $C\#$, onde foram executados numa máquina virtual fornecida pela plataforma *Google Cloud* numa máquina equipada com CPU Intel Sandy Bridge com clock 3.3 Ghz, 12GB de memória RAM e Windows 7 64 bits.

Na primeira busca o algoritmo encontrou um contra-exemplo para a desigualdade $(*_1)$: o cografo que denotaremos por G_a, ilustrado na Figura 4, possui as seguintes características: $|V| = 14$; $\lambda(G_a) \approx 6,9618$; $\chi_b(G) = m(G) = 8$; $\chi_b(G) - (\lambda(G) + 1) = 0,038$.

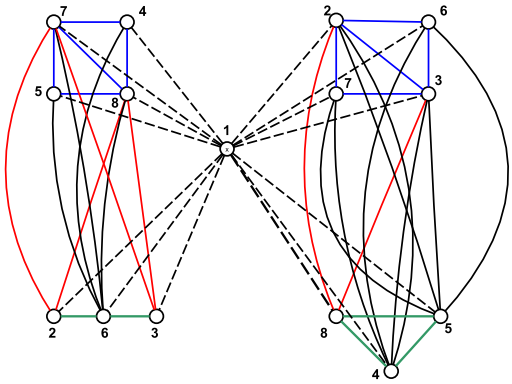

Figura 4: Contra-exemplo G_a

Então a afirmação $(*_1)$ proposta é falsa.

5.0.1. Formulação de uma nova afirmação

Com o contra-exemplo encontrado, observamos que G_a nega a desigualdade da afirmação por menos de um décimo, o que nos motiva a introdução de uma nova afirmação:

Todo cografo conexo G satisfaz $\chi_b(G) \leq \lceil \lambda(G) \rceil + 1$ **$(*_2)$,**
onde usamos o parâmetro $\lceil \lambda(G) \rceil$ ao invés de $\lambda(G)$.

Nesta reformulação o grafo G_a não é mais um contra-exemplo.

Uma forma eficiente de calcular o teto do índice de um grafo é usar o algoritmo de diagonalização e fazer uso do seguinte teorema para otimizar o processo.

Teorema 5.1. *Cvetković e Rowlinson [1990] Todo grafo G satisfaz $\delta \leq \overline{d} \leq \lambda(G) \leq \Delta$, onde δ, $\overline{d}$ e Δ são os graus mínimo, médio e máximo de G, nesta ordem.*

Algorithm 5.6 Cálculo do teto do índice

Input: Coárvore T do cografo G.
Output: Valor $\lceil \lambda(G) \rceil$.

```
1: procedure TetoDoIndice(T)
2:     Seja k = ⌈d̄(G)⌉
3:     while IndiceMaiorQue(T, k) do
4:         k ← k + 1
5:     end while
6:     return k
7: end function
```

De forma semelhante ao feito no algoritmo , podemos efetuar procedimento semelhante para validar a afirmação $(*_2)$.

Antes de verificar a afirmação $(*_2)$, observamos que o parâmetro m-grau poderia assumir o papel do número b-cromático nesta afirmação, uma vez que como visto na Proposição 2.1, todo grafo G satisfaz a desigualdade $\chi_b(G) \leq m(G)$. Então propomos a seguinte relação para todo cografo conexo:

$$m(G) \leq \lceil \lambda(G) \rceil + 1 \qquad (*_3)$$

Se a desigualdade $(*_3)$ for mantida, obteremos trivialmente a veracidade da afirmação $(*_2)$.

O parâmetro m-grau é mais simples de trabalhar, inclusive computacionalmente, uma vez que sua definição envolve apenas a sequência de graus do grafo.

Algorithm 5.7 Verificação das desigualdades $(*_2)$ e $(*_3)$

Output: Grafo contraexemplo

```
1:  procedure ValidacaoDesigualdadeMgrau
2:      n = 2
3:      while true do
4:          Seja T a menor coárvore com n vértices
5:          repeat
6:              if Mgrau(T) > TetoDoIndice(T) + 1 then
7:                  If Bcromatico(T) > TetoDoIndice(T) + 1 then stop
8:              end if
9:              T ← ArvoreSeguinte(T)
10:         until T = null
11:         n ← n + 1
12:     end while
13: end function
```

E alguns contra-exemplos para a desigualdade $(*_3)$ foram encontrados. O primeiro foi o grafo $G_b := \overline{K_7} \vee (K_6 \cup \overline{K_6})$, que possui 19 vértices, cujas características são: $\lceil \lambda(G_b) \rceil = 11$; $m(G) = 13$; $\chi(G) = \chi_b(G) = 7$.

Como podemos gerar os cografos com número fixado de vértices, podemos naturalmente efetuar a contagem de quantos cografos existem. Após realizados os testes constatamos que a proporção de cografos que não satisfazem apenas a desigualdade $(*_3)$ é insignificante, isto é, aqueles cografos que acusaram **false** na linha (6) e **true** na linha (7), do Algoritmo 5.7. No final do texto segue a Tabela 1 com o número de cografos conexos existentes para cada número de vértices fixado, o número de cografos que não stisfazem apenas $(*_3)$, a proporção e o tempo de execução do programa.

Em contra-partida, executamos a validação da afirmação $(*_2)$ e obtivemos ótimos resultados: não foi encontrado nenhum contra-exemplo para esta desigualdade dentre todos os cografos conexos até 21 vértices, contabilizando mais de 1 bilhão de grafos. Vale observar que devido à enorme quantidade de cografos existentes, o programa se manteve em execução por 14 dias. Então segue o resultado cuja demonstração foi feita computacionalmente.

Proposição 5.1. *Todos os cografos conexos de até 21 vértices satisfazem* $\chi_b(G) \leq \lceil \lambda(G) \rceil + 1$.

E propomos a seguinte conjectura:

CONJECTURA 5.1. *Todo cografo conexo G satisfaz* $\chi_b(G) \leq \lceil \lambda(G) \rceil + 1$.

Observe que esta é uma cota razoável, uma vez que a igualdade vale para o grafo completo, enquanto pode se afastar em alguns casos, como no exemplo do cografo esboçado abaixo, que possui número b-cromático 3 e índice aproximadamente $4,9$.

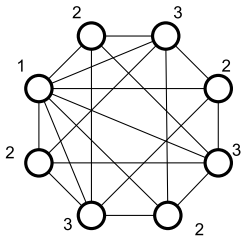

Figura 5: cografo G

Vértices	Qtd. cografos conexos	Não vale desigualdade $(*_3)$	proporção	Tempo
2	1	0	0 %	$<$ 0,1s
3	2	0	0 %	$<$ 1s
4	5	0	0 %	$<$ 1s
5	12	0	0 %	$<$ 1s
6	33	0	0 %	$<$ 1s
7	90	0	0 %	$<$ 1s
8	261	0	0 %	$<$ 1s
9	766	0	0 %	$<$ 1s
10	2312	0	0 %	$<$ 1s
11	7068	0	0 %	$<$ 1s
12	21965	0	0 %	$<$ 1min
13	68954	0	0 %	$<$ 1min
14	218751	0	0 %	$<$ 1min
15	699534	0	0 %	13 min
16	2253676	0	0 %	50 min
17	7305788	0	0 %	3h
18	23816743	0	0 %	9h
19	78023602	3	$3,845 \times 10^{-8}$ %	30h
20	256738751	21	$8,179 \times 10^{-8}$ %	3 dias
21	848152864	88	$1,037 \times 10^{-7}$ %	9 dias

Tabela 1: Geração de cografos e verificação da desigualde $(*_3)$

Referências

Alkhateeb, M. e Kohl, A. (2011). Upper bounds on the b-chromatic number and results for restricted graph classes. *Discussiones Mathematicae Graph Theory*, 31(4):709–735.

Bonomo, F., Duran, G., Maffray, F., Marenco, J., e Valencia-Pabon, M. (2009). On the b-coloring of cographs and p4-sparse graphs. *Graphs and Combinatorics*, 25(2):153–167.

Corneil, D. G., Lerchs, H., e Burlingham, L. S. (1981). Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163–174.

Corneil, D., Perl, Y., e Stewart, L. (1984). Cographs: recognition, applications and algorithms. *Congressus Numer*, 43:249–258.

Cvetkovic, D. (1972). Chromatic number and the spectrum of a graph. *Publ. Inst. Math.(Beograd)*, 14(28):25–38.

Cvetković, D. e Rowlinson, P. (1990). The largest eigenvalue of a graph: A survey. *Linear and multilinear algebra*, 28(1-2):3–33.

Irving, R. W. e Manlove, D. F. (1999). The b-chromatic number of a graph. *Discrete Applied Mathematics*, 91(1):127–141.

Jacobs, D. P., Trevisan, V., e Tura, F. C. (2017). Eigenvalue location in cographs. *Discrete Applied Mathematics*.

Jones, A. A., Protti, F., e Del-Vecchio, R. R. (2016). Gerador de cografos com atraso linear. In *Anais do XLIVII Simpósio Brasileiro de Pesquisa Operacional, Vitória - ES*, p. 3123–3134.

Kouider, M. e Mahéo, M. (2002). Some bounds for the b-chromatic number of a graph. *Discrete Mathematics*, 256(1):267–277.

Wilf, H. S. (1967). The eigenvalues of a graph and its chromatic number. *J. London Math. Soc*, 42 (1967):330.