**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**
**DEPARTAMENTO ACADÊMICO DE INFORMÁTICA**
**BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**CINTIA IZABEL CARARO**

# COLORAÇÃO BICLIQUE EM COGRAFOS

**TRABALHO DE CONCLUSÃO DE CURSO**

**PONTA GROSSA**
**2018**

**CINTIA IZABEL CARARO**

# COLORAÇÃO BICLIQUE EM COGRAFOS

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientadora: Prof[a]. Dr[a]. Sheila Morais de Almeida

**PONTA GROSSA**

**2018**

Ministério da Educação
**Universidade Tecnológica Federal do Paraná**
Câmpus Ponta Grossa
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Bacharelado em Ciência da Computação

# TERMO DE APROVAÇÃO

COLORAÇÃO BICLIQUE EM COGRAFOS

por

CINTIA IZABEL CARARO

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em **12 de junho de 2018** como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. A candidata foi arguida pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho **aprovado**.

_____
Profa. Dra. Sheila Morais de Almeida
Orientadora

_____
Prof. MSc. Clayton Kossoski
Membro titular

_____
Profa. Dra. Simone Nasser Matos
Membro titular

_____
Prfa. Dra. Helyane Bronoski Borge
Profa. Responsável pelo
Trabalho de Conclusão de Curso

_____
Prof. MSc. Saulo Jorge Beltrão de Queiroz
Coordenador do curso

- A Folha de Aprovação assinada encontra-se arquivada na Secretaria Acadêmica -

## AGRADECIMENTOS

Primeiro agradeço a Divina Providência por tudo ter providenciado e ao Espírito Santo por ter tudo feito.

Agradeço aos meus pais Verônica e Noel por me ensinarem valores que norteiam minha vida. Aos meus irmãos Solange, Samoel, Selma, César e Raquel por sempre estarem dispostos a me ajudar. Agradeço a minha irmã Simone e meu cunhado José, por todo apoio que expressaram desde a escolha do curso, por me acolherem em sua casa, por todas as caronas, por sempre me incentivarem.

De forma especial, agradeço a minha orientadora Sheila Morais de Almeida pela dedicação e paciência em todos os momentos.

*“Por vezes, sentimos que aquilo que fazemos não é,*
*senão, uma gota de água no mar.*
*Mas o mar seria menor se lhe faltasse uma gota.”*
*Santa Teresa de Calcutá*

## RESUMO

CARARO, Cintia Izabel. **Coloração Biclique em Cografos**. 2018. 59 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2018.

Uma $k$-coloração biclique é a atribuição de $k$ cores aos vértices de um grafo de modo que nenhum subgrafo induzido bipartido completo maximal seja monocromático. Decidir se um grafo tem uma $k$-coloração biclique para um dado $k$ é um problema $\Sigma_2^P$-completo. Este trabalho apresenta solução em tempo linear para o Problema da Coloração Biclique em uma subclasse dos cografos. Além disso, são apresentados novos algoritmos para enumeração de conjuntos independentes maximais, enumeração de bicliques e contagem de bicliques em cografos.



**Palavras-chaves: Biclique. Cografo. Coloração.**

**ABSTRACT**

CARARO, Cintia Izabel. **Biclique Coloring Cographs**. 2018. 59 p. Work of Conclusion Course (Computer Science Bachelor) - Federal University of Technology - Paraná. Ponta Grossa, 2018.

A $k$-biclique coloring is an assignment of $k$ colors to the vertices of a graph so that its maximal bipartite induced subgraphs are not monochromatic. To decide if a graph has a $k$-biclique coloring for a given $k$ is a $\Sigma_2^P$-complete problem. This paper presents a linear solution to the Biclique Coloring Problem in a subclass of cographs. Furthermore, new algorithms to enumerate the maximal independent sets and bicliques and to count bicliques in cographs are presented.

**Key-words: Biclique. Cograph. Coloring.**

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| acp | Ancestral comum mais próximo |
| *Cograph* | *Complement Reducible Graph* (Grafo Redutível por Complemento) |
| LexBFS | *Lexicographic Breadth-First Search* (Busca em Largura Lexicográfica) |
| LexBFS$^-$ | *Lexicographic Breadth-First Search minus* (Busca em Largura Lexicográfica *minus*) |
| PSV | Propriedade do Subconjunto da Vizinhança |

## LISTA DE SÍMBOLOS

| | |
|---|---|
| $V(G)$ | Conjunto de vértices de um grafo $G$ |
| $E(G)$ | Conjunto de arestas de um grafo $G$ |
| $C_n$ | Ciclo com $n$ vértices |
| $K_{\|A\|,\|B\|}$ | Grafo bipartido completo com partições de tamanho $A$ e $B$ |
| $\{A, B\}$ | Subconjuntos de vértices $A$ e $B$ de um $K_{\|A\|,\|B\|}$ |
| $K_n$ | Grafo completo com $n$ vértices |
| $P_n$ | Caminho com $n$ vértices |
| $T(w)$ | Árvore enraizada em um vértice $w$ |
| $\chi(G)$ | Número cromático do grafo $G$ |
| $\chi_{bc}(G)$ | Número biclique cromático do grafo $G$ |
| $A \cup B$ | União entre os conjuntos $A$ e $B$ |
| $A * A$ | Junção entre os conjuntos $A$ e $B$ |
| $\overline{G}$ | Complemento de um grafo $G$ |
| $N(v)$ | Vizinhança aberta de um vértice $v$ |
| $N[v]$ | Vizinhança fechada de um vértice $v$ |
| $\beta(G)$ | Número de bloco de $G$ |
| $B_i$ | Conjunto de vértices sequenciais universais em uma ordenação *threshold* |
| $I_i$ | Conjunto de vértices sequenciais isolados em uma ordenação *threshold* |

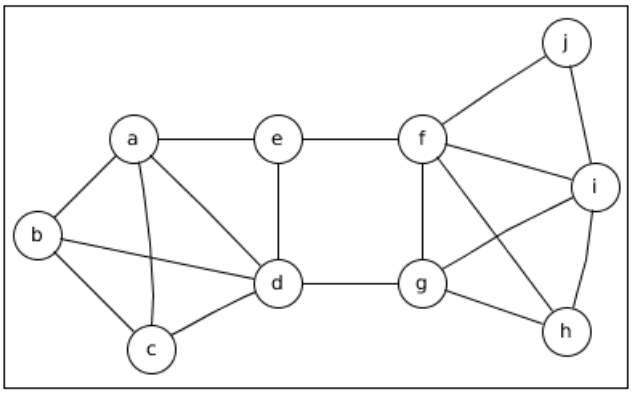
**SUMÁRIO**

# 1 INTRODUÇÃO

O Problema da Coloração de Vértices teve seu início quando Alfred Bray Kempe em 1989 modelou o Problema das Quatro Cores como um grafo para tentar resolvê-lo. Esse problema consistia em provar uma conjectura, criada por Frederick Guthrie, na qual qualquer mapa poderia ser pintado com quatro cores de modo que regiões vizinhas não possuíssem a mesma cor. No grafo $G = (V(G), E(G))$, com conjunto de vértices $V(G)$ e conjunto de arestas $E(G)$, modelado por Kempe, cada região é um vértice $v \in V(G)$ e, se duas regiões $x$ e $y$ são vizinhas, existe uma aresta entre os vértices que as representam, ou seja, $(x, y) \in E(G)$. Dessa forma, Kempe transformou o Problema das Quatro Cores no problema de colorir os vértices do grafo - que representa um mapa - com no máximo quatro cores, sendo que vértices adjacentes não podem possuir a mesma cor. Para mais detalhes sobre a história do Problema das Quatro Cores, uma boa introdução é apresentada por Chartrand e Zhang (2008).

Essa forma de coloração - onde vizinhos não podem ter a mesma cor - é chamada de coloração própria. Várias formas de coloração em grafos surgiram desde então para resolver diferentes problemas onde o objetivo é a otimização no uso de recursos. Um desses problemas é usar o menor número de cores para colorir um grafo de forma que certas estruturas não sejam monocromáticas. Uma dessas estruturas é a das cliques maximais. Uma clique em um grafo é um subconjunto de vértices que são todos adjacentes entre si. A clique é maximal se não está contida em outra clique. Considerando o problema de colorir os vértices de forma que cliques maximais não sejam monocromáticas, qual o menor número de cores necessárias para colorir os vértices da Figura 1? As cliques maximais do grafo são $\{a, b, c, d\}$, $\{a, d, e\}$, $\{e, f\}$, $\{g, f, h, i\}$, $\{f, j, i\}$, $\{d, g\}$.

**Figura 1 – Grafo com seis cliques maximais**

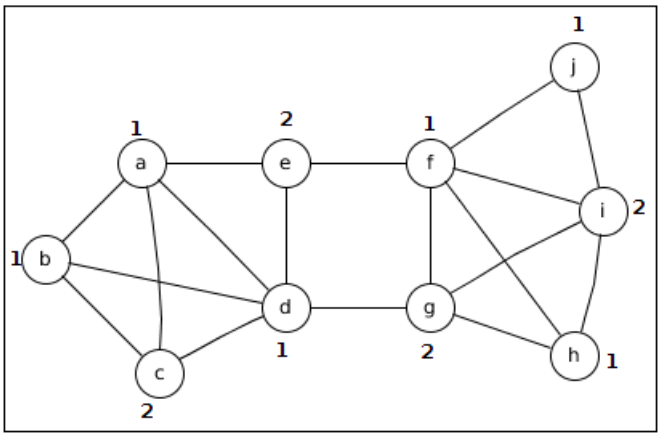



**Fonte: Autoria própria**

Uma coloração de vértices na qual as estruturas que não podem ser monocromáticas são cliques é chamada de coloração clique. Nesse trabalho as cores são representadas como números. A Figura 2 apresenta uma coloração clique mínima para o grafo da Figura 1. Observe que toda clique maximal tem vértices coloridos com as cores $1$ e $2$. Como não é possível colorir as cliques

maximais de um grafo com menos de duas cores de forma que não sejam monocromáticas, a coloração clique apresentada na Figura 2 é ótima, ou seja, usa o mínimo possível de cores. O problema de colorir os vértices de forma que cliques maximais não sejam monocromáticas usando o mínimo de cores é conhecido como Problema da Coloração Clique.

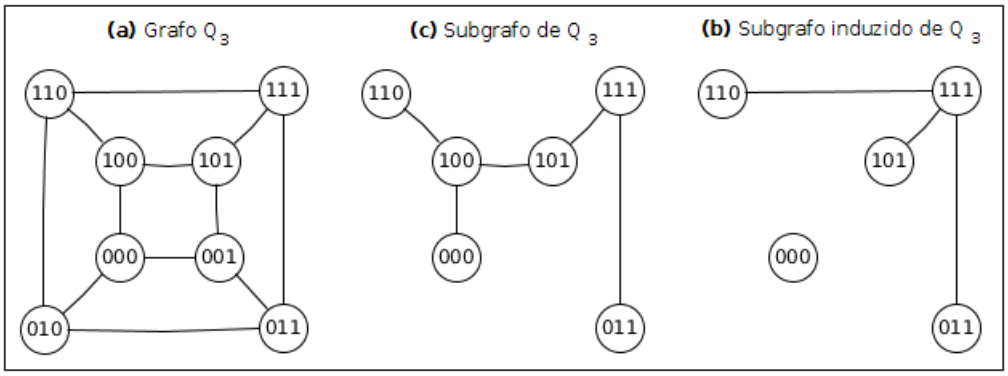
**Figura 2 – Exemplo coloração clique**




**Fonte: Autoria própria**

Em geral, as estruturas que não podem ser monocromáticas são subgrafos induzidos. Um subgrafo induzido de um grafo $G$ é qualquer grafo com subconjunto de vértices $S \subseteq V(G)$ que contém todas as arestas de $E(G)$ com ambos os extremos em $S$. Para apresentar exemplos de subgrafos induzidos, vamos considerar os grafos $k$-cubo. Um $k$-cubo, denotado por $Q_k$, é um grafo cujo conjunto de vértices é o conjunto de todas as *strings* com $k$ bits e dois vértices são adjacentes se, e somente se, diferem em exatamente um bit. Na Figura 3(b) é apresentado um subgrafo de $Q_3$ induzido pelo subconjunto de vértices $S = \{110, 111, 101, 000, 011\}$. Perceba que todas as arestas que existem em $E(Q_3)$ entre os vértices de $S$ são preservadas em $Q'$. Na Figura 3(c) temos um subgrafo de $Q_3$ que não é induzido, isso se deve a aresta $(110, 111)$ existir em $E(Q_3)$ mas não em $E(Q'')$.

**Figura 3 – Grafo $Q_3$ e, respectivamente, um subgrafo e um subgrafo induzido de $Q_3$**

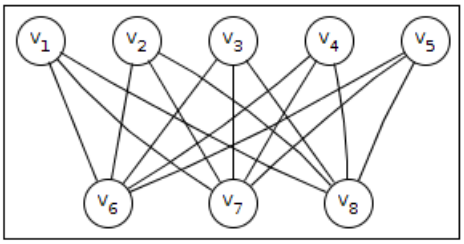



**Fonte: Autoria própria**

Uma coloração introduzida por Terlisky (2010) considera que as estruturas que não podem ser monocromáticas na coloração de vértices são subgrafos induzidos bipartidos completos.

Um grafo é bipartido se seu conjunto de vértices pode ser particionado em dois subconjuntos onde em cada subconjunto os vértices são dois a dois não adjacentes. Conjuntos de vértices que não são adjacentes entre si são chamados de conjuntos independentes. Considere um grafo bipartido $G$ com partição do conjunto de vértices $[A, B]$. Se existe aresta de cada vértice de $A$ para todos os vértices de $B$, então $G$ é um grafo bipartido completo, denotado por $K_{|A|,|B|}$. A Figura 4 apresenta um exemplo de grafo bipartido completo cujas partes têm tamanhos 5 e 3.

**Figura 4 – Bipartido completo** $K_{5,3}$




**Fonte: Autoria própria**

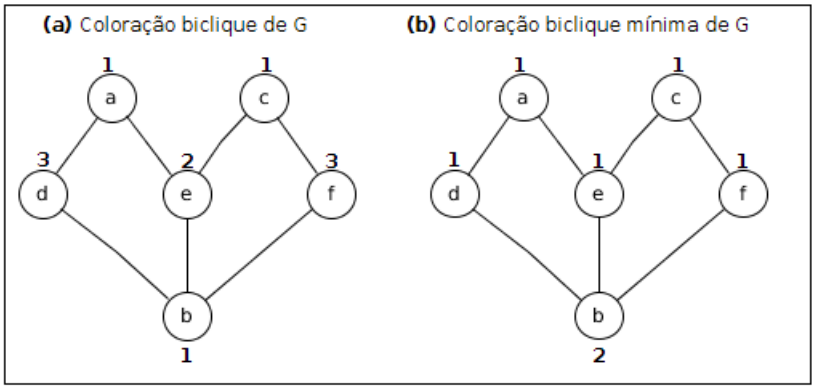
Um conjunto de vértices que induz um subgrafo bipartido completo é chamado de biclique. Se $S$ é uma biclique de um grafo $G$ que não está contida em outra biclique, então $S$ é uma biclique maximal. Nesse trabalho, toda biclique é maximal, a não ser que seja dito o contrário. Considerando o grafo da Figura 4 como exemplo, os conjuntos independentes da biclique, serão denotados por: $\{\{v_1, v_2, v_3, v_4, v_5\}, \{v_6, v_7, v_8\}\}$, onde $\{v_1, v_2, v_3, v_4, v_5\} \in A$ e $\{v_6, v_7, v_8\} \in B$.

Uma coloração biclique de $G$ é uma função que associa uma cor de um conjunto com $k$ cores a cada vértice de $G$, de forma que qualquer biclique maximal não seja monocromática. O Problema da Coloração Biclique consiste em encontrar o menor $k$ para o qual $G$ possui uma coloração biclique. Esse número é chamado de número biclique cromático de $G$ e denotado por $\chi_{bc}(G)$ (TERLISKY, 2010).

No grafo $G$ da Figura 5 há quatro bicliques maximais: $\{a, b, d, e\}$, $\{b, c, e, f\}$, $\{e, a, c, b\}$ e $\{b, d, e, f\}$. Em 5(a) é apresentada uma coloração de vértices própria para o grafo resultando em uma coloração biclique válida - como cada par de vértices adjacentes possui cor distinta toda biclique é policromática. Em 5(b) é apresentada uma coloração biclique mínima para o grafo: como o vértice $b$ é comum a todas as bicliques basta que ele tenha cor distinta para a coloração ser biclique, assim $\chi_{bc}(G) = 2$.

Ao introduzir o Problema da Coloração Biclique, Terlisky (2010) determinou que se $G$ é um grafo bipartido, então $\chi_{bc}(G) = 2$; se $K_n$ é um grafo completo com $n$ vértices, então $\chi_{bc}(K_n) = n$; e se $C_n$ é um ciclo com $n$ vértices e $n > 3$, então $\chi_{bc}(C_n) = 2$. Além dessas classes, Terlisky (2010) apresentou algoritmos polinomiais para: alguns casos dos grafos *split*; grafos *threshold*; grafos cordais *diamond-free* e grafos onde cada aresta pertence a uma única biclique. Outros resultados conhecidos são a solução do Problema da Coloração Biclique em tempo polinomial para para grafos potência (FILHO *et al.*, 2015) e para grafos *unichord-free* (FILHO; MACHADO; FIGUEIREDO, 2017).

**Figura 5 – Coloração biclique válida e mínima de $G$**




**Fonte: Autoria própria**

Para entender a classe de grafos investigada nesse trabalho é necessário conhecer o conceito de caminho. Um caminho é um subgrafo de um grafo $G$, não necessariamente induzido, com subconjunto de vértices $S \subseteq V(G)$ que podem ser rotulados em sequência $(v_1, v_2, v_3, \ldots, v_k)$ de tal forma que $(v_i, v_{i+1}) \subseteq E(G)$ para todo $i$ tal que $1 \leq i \leq k-1$. Quando os únicos vértices e arestas do grafo $G$ são exatamente aqueles que compõem o caminho, então $G$ é um grafo caminho e é denotado por $P_n$ - um grafo caminho com $n$ vértices.

Nesse trabalho, o Problema da Coloração Biclique é investigado na classe dos cografos, que são grafos que não possuem $P_4$ como subgrafo induzido. Foi desenvolvida a primeira técnica para enumeração de bicliques em cografos. Também é apresentado um algoritmo linear para coloração biclique ótima de um subconjunto dos cografos.

## 1.1 MOTIVAÇÃO E JUSTIFICATIVA

Na computação, existem dois tipos de problemas comuns: os problemas de otimização e os de decisão. Problemas de otimização caracterizam-se por levantarem questões sobre qual o número mínimo (nos problemas de minimização) ou máximo (nos problemas de maximização) de recursos para resolver o problema. Por exemplo, questionar qual o número mínimo de cores com que se pode obter uma coloração biclique de um dado grafo $G$ é um problema de otimização, mais especificamente, um problema de minimização. Para cada problema de otimização, existe uma versão de decisão. Na versão de decisão, a pergunta é ligeiramente diferente. Para o caso geral, considerando uma instância do problema e um dado número $k$, a pergunta que se faz é se é possível resolver o problema utilizando apenas $k$ unidades de recursos. Tais problemas chamam-se problemas de decisão, pois a resposta é apenas “sim” ou “não”.

Observa-se que uma vez que se encontre algoritmo eficiente para resolver a versão de decisão de um determinado problema, pode-se executá-lo em um laço de repetição que itera alterando o valor de $k$ entre valores mínimo e máximo possíveis, para determinar a resposta do problema de otimização. Por exemplo, se conhecermos um algoritmo eficiente que resolve a

versão de decisão do Problema da Coloração Biclique, então podemos criar um laço de repetição que itera variando o valor de $k$ entre $1$ e $n$, onde $n$ é a quantidade de vértices, e para cada valor de $k$ executamos o algoritmo que resolve a versão de decisão. A primeira vez que o algoritmo responder "sim", retorna-se o valor de $k$. Então, se o número de possibilidades para o valor de $k$ é polinomial em função do tamanho da entrada do problema, isso implica que ao encontrar um algoritmo eficiente que resolva a versão de decisão de um problema, encontra-se também um algoritmo eficiente para resolver a versão de otimização do mesmo problema.

Diante disso, as versões de decisão dos problemas computacionais foram divididas em classes de complexidade. A classe de complexidade P é a dos problemas para os quais se pode obter algoritmos eficientes, também chamados de algoritmos polinomiais. Um algoritmo é eficiente se consegue resolver qualquer instância do problema utilizando um número de instruções limitado superiormente por um polinômio em função do tamanho da entrada do problema. Como exemplo pode-se citar os algoritmos capazes de ordenar um conjunto de $n$ números utilizando uma quantidade de instruções computacionais não superior a $cn^2$, onde $c$ é uma constante positiva e $n$ é a quantidade de números a serem ordenados (o tamanho da entrada).

Em alguns casos, não se sabe se o problema pertence ou não à classe P. Suponha que existe um oráculo capaz de resolver tais problemas instantaneamente e responder "sim" ou "não". Suponha que quando esse oráculo apresenta a resposta, ele também oferece um certificado que garante a veracidade da resposta. Se, nos casos em que a resposta é "sim", for possível construir um algoritmo polinomial em função do tamanho da entrada do problema para verificar se o certificado é válido, então o problema pertence à classe NP. Por exemplo, a versão de decisão do Problema da Coloração de Vértices é responder se é possível colorir os vértices de um grafo com $k$ cores de forma que vértices vizinhos tenham cores distintas. Não se conhecem algoritmos eficientes que resolvam esse problema. Mas, uma vez que um oráculo responda que é possível colorir os vértices do grafo com $k$ cores de forma que vizinhos tenham cores diferentes e apresente a tal coloração como certificado, pode-se construir um algoritmo eficiente que verifica se a coloração está correta, ou seja, se não há vizinhos com a mesma cor e se foram usadas $k$ cores. Então, o Problema da Coloração de Vértices pertence à classe de complexidade NP. Mais que isso, se houver um algoritmo eficiente que resolva o Problema da Coloração de vértices, então existe algoritmo eficiente para todos os problemas da classe de complexidade NP (DAILEY, 1980). Problemas NP para os quais a existência de algoritmos eficientes implica na existência de algoritmos eficientes para todos os problemas NP são chamados de NP-completos.

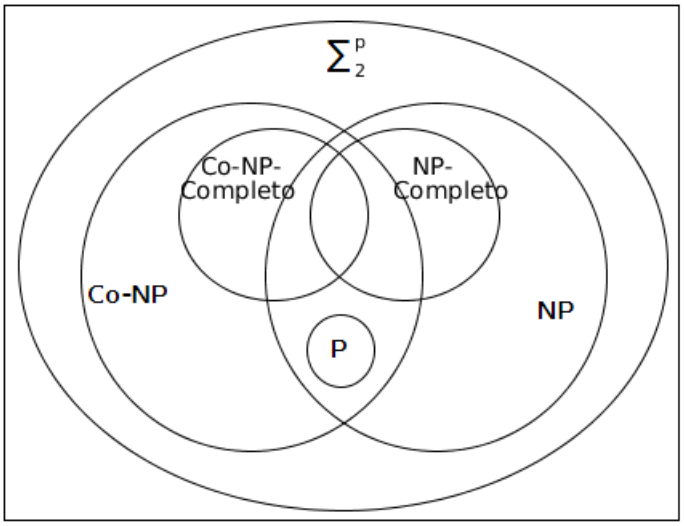
Não se sabe se o Problema da Coloração Biclique pertence a NP. O que se sabe é que uma vez que o oráculo apresente resposta "sim' com o respectivo certificado, verificar se este certificado é válido é um problema em NP (GROSHAUS; SOULIGNAC; TERLISKY, 2012). Esse fato coloca o Problema da Coloração Biclique em uma classe de complexidade maior que NP, conhecida como $\Sigma_2^P$ tornando-o teoricamente um problema mais difícil de se resolver. Mais que isso, sabe-se que o Problema da Coloração Biclique é $\Sigma_2^P$-completo (GROSHAUS; SOULIGNAC; TERLISKY, 2012), o que significa que uma solução eficiente para o mesmo implica

em solução eficiente para todos os outros problemas desta classe de complexidade.

Agora, suponha que existe um algoritmo polinomial para verificar se um certificado dado pelo oráculo é válido quando a resposta para o problema é “não”. Então o problema pertence à classe co-NP. Um exemplo de problema na interseção das classes P, NP e co-NP é o de determinar se um dado grafo é bipartido. Existe algoritmo polinomial que verifica se um dado grafo $G$ é bipartido. Então esse problema pertence à classe P. Como já visto, um grafo é bipartido se, e somente se, seu conjunto de vértices pode ser particionado em dois conjuntos independentes (vértices do mesmo subconjunto não são vizinhos). Um oráculo pode apresentar uma partição do conjunto de vértices em conjuntos independentes quando responde “sim”. Suponha que o oráculo apresentou como certificado uma partição do conjunto de vértices em $A$ e $B$. Então, pode-se construir um algoritmo para checar se cada aresta de $G$ tem um vértice em $A$ e outro em $B$. Isso pode ser feito em tempo polinomial. Então, esse problema pertence à NP. Sabe-se que um grafo é bipartido se, e somente se, não possui ciclo ímpar. Então, quando um oráculo responde “não”, ele pode apresentar um ciclo ímpar do grafo como certificado. Verificar se o ciclo ímpar apresentado é um subgrafo de $G$ pode ser feito por um algoritmo polinomial. Então, esse problema também está em co-NP (BONDY; MURTY, 2008).

De forma geral, P $\subseteq$ NP $\cap$ co-NP, como representado na Figura 6. Para cada par de classes da Figura 6 não existe prova se a relação de contigência é estrita, ou seja, pode ser que P = NP, NP = co-NP e até mesmo $\sum_2^P$ = P.

**Figura 6 – Relação entre as classes de complexidade**




**Fonte: Autoria própria**

Sabe-se que o Problema da Coloração Biclique pertence à classe co-NP (FILHO *et al.*, 2015). Um problema é co-NP-completo se a existência de um algoritmo que o resolva em tempo polinomial implicar na existência de algoritmos polinomiais que resolvam todos os problemas da classe co-NP. O Problema da Coloração Biclique é um problema co-NP-completo (FILHO *et al.*, 2015).

Uma importante questão em aberto da Teoria da Computação é se as classes de pro-

blemas NP e co-NP são iguais. Caso a resposta seja positiva, todo problema NP pode ser reduzido ao Problema da Coloração Biclique. Neste caso, um algoritmo polinomial que resolva o Problema da k-coloração biclique implica em resolver de forma computacionalmente eficiente todos os problemas em NP, incluindo alguns problemas famosos, como o Problema do Caixeiro Viajante, da Coloração de Vértices e da Fatoração. Tal resultado teria impacto significativo na sociedade ao implicar que todas mensagens cifradas com criptografia RSA poderiam ser decifradas em tempo eficiente e que vários problemas de otimização com larga aplicabilidade no setor produtivo, para os quais não se conhecem algoritmos eficientes, poderiam ser resolvidos em tempo polinomial.

O conhecimento sobre as bicliques de um grafo é importante na resolução de problemas de diversas áreas como inteligência artificial (WILLE, 1982), linguagens formais e autômatos (FROIDURE, 1995), ordens parciais (LUBIW, 1990) e na exploração de comunidades em redes sociais (KUMAR *et al.*, 1999). Na medicina, há aplicações utilizando enumeração de bicliques (NAGARAJAN; KINGSFORD, 2008) e também no ramo da genética (ATLURI *et al.*, 2000). Aplicações de listagem de bicliques em grafos de iteração proteína-proteína, no ramo da biologia, foram estudadas por Bu *et al.* (2003) e Li *et al.* (2007).

Em relação à classe de grafos escolhida para se estudar o Problema da Coloração Biclique, sabe-se que para os cografos muitos problemas NP-completos no caso geral, como o Problema da Coloração de Vértices, Problema da Clique Máxima, Problema do Ciclo Hamiltoniano, possuem solução eficiente quando restritos aos cografos (BRANDSTÄDT; LE; SPINRAD, 1999). Esse fato é um indício de que problemas computacionalmente difíceis possam ser resolvidos para esta classe.

Os cografos possuem uma estrutura que apresenta informações sobre as adjacências do grafo. Nesse trabalho, mostramos que por meio dessa estrutura é possível identificar e enumerar todas as bicliques. Além disso, em alguns casos provamos que é possível colori-las de forma eficiente.

Além disso, embora a maioria dos grafos não seja um cografo, qualquer grafo pode ser transformado em um cografo pelo acréscimo de algumas arestas de forma que o grafo não tenha mais $P_4$ induzido (CORNEIL; PERL; STEWART, 1985). Talvez, por essa propriedade, a solução do Problema da Coloração Biclique para os cografos possa inspirar formas de resolver o problema em outros subconjuntos de grafos: escolha um grafo qualquer, insira arestas para torná-lo um cografo, resolva o Problema da Coloração Biclique, remova as arestas inseridas e tente responder à seguinte questão: esta é uma solução ótima para o grafo dado? É interessante questionar em que casos a resposta será positiva.

## 1.2 ORGANIZAÇÃO DO TRABALHO

O Capítulo 2 apresenta propriedades dos cografos. No Capítulo 3 é apresentado o Problema da Enumeração de Bicliques. Esse problema consiste em enumerar todas as bicliques de um dado grafo $G$. Segundo Eppstein (1994), não é possível resolver esse problema em tempo polinomial em relação ao tamanho da entrada, uma vez que o número de bicliques de $G$ pode ser exponencial em relação ao número de vértices. Prisner (2000) estabeleceu o limitante superior mais justo conhecido para o número de bicliques em um grafo qualquer como $n^{5/2}(1.618034)^n + O(1)$.

Embora seja necessário saber como se estruturam as bicliques para colorir um grafo em coloração biclique, não é necessário resolver o Problema da Enumeração de Bicliques para resolução do Problema da Coloração Biclique. Um exemplo disso é o caso dos bipartidos, onde o Problema da Enumeração de Bicliques não pode ser resolvido em tempo polinomial, pois o número de bicliques pode chegar a ser $2^{n/2} \approx 1.41^n$ (PRISNER, 2000), já a coloração biclique é trivial (TERLISKY, 2010).

As características da coloração biclique são detalhados no Capítulo 4. Nele também são apresentadas soluções existentes para o Problema da Coloração Biclique em subclasses dos cografos. Essas subclasses são: grafos completos e grafos *threshold* - cografos que podem ter seu subconjunto de vértices particionado em uma clique e um conjunto independente.

No Capítulo 5 são apresentados novos resultados em coloração biclique para cografos. No capítulo seguinte é feita a conclusão do trabalho resumindo os resultados alcançados e propostos estudos futuros.

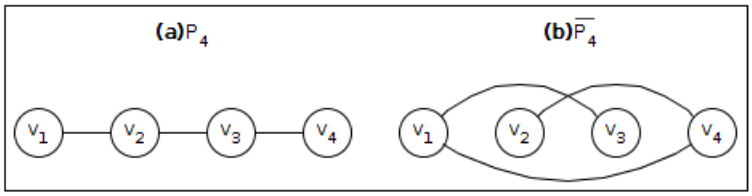
# 2 COGRAFOS

O termo cografos foi introduzido por Lerchs (1971) e vem do inglês *Cograph - complement reducible graph* (grafos redutíveis por complemento). São assim chamados por poderem ser reduzidos a vértices isolados por sucessivas aplicações de operações de complemento em suas componentes.

O resultado da operação de complemento em um grafo $G$, é um grafo $\overline{G}$ tal que dois vértices são adjacentes em $\overline{G}$ se, e somente se, são distintos e não são adjacentes em $G$, ou seja, $V(\overline{G}) = V(G)$ e $E(\overline{G}) = \{v_iv_j : v_i \in V(G) \wedge v_j \in V(G) \wedge i \neq j \wedge v_iv_j \notin E(G)\}$.

Por consequência da definição de grafos redutíveis por complemento, os cografos também foram caraterizados como grafos que não possuem um $P_4$ como subgrafo induzido. Na Figura 7 é apresentado um grafo $P_4$ em (a) e o seu complemento $\overline{P_4}$ em (b). Observe que o complemento do grafo $P_4$ também é um grafo $P_4$ com caminho dado pela sequência $(v_2, v_4, v_1, v_3)$. Por isso, o grafo $P_4$ é dito auto-complementar e não pode ser reduzido a vértices isolados por operação de complemento, contrariando a definição de cografo.

**Figura 7 – Grafo caminho com $4$ vértices e seu complemento**




**Fonte: Autoria própria**

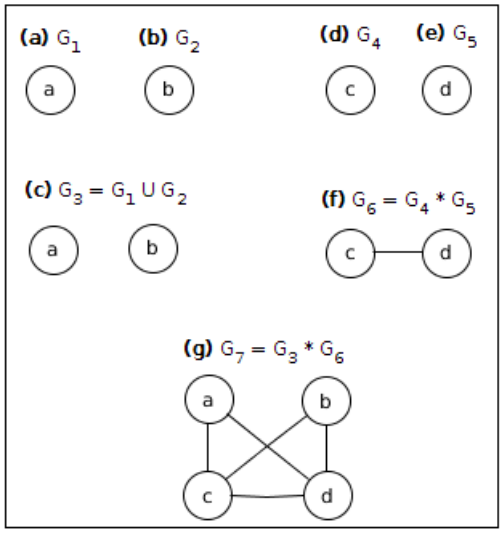
Os cografos também podem ser definidos de forma construtiva por meio de sucessivas aplicações de união e junção de grafos partindo de um $K_1$. Na operação de união entre dois grafos $G_1$ e $G_2$ dados, o grafo resultante, $G_3 = G_1 \cup G_2$, possui os mesmos vértices e arestas dos grafos envolvidos na operação, isto é, $V(G_3) = V(G_1) \cup V(G_2)$ e $E(G_3) = E(G_1) \cup E(G_2)$. Já na operação de junção, além dos vértices e arestas dos grafos $G_1$ e $G_2$, o grafo resultante $G_3 = G_1 * G_2$ possui arestas ligando cada vértice de $G_1$ a todos os vértices de $G_2$, ou seja, $V(G_3) = V(G_1) \cup V(G_2)$ e $E(G_3) = E(G_1) \cup E(G_2) \cup \{v_iv_j : v_i \in V(G_1)$ e $v_j \in V(G_2)\}$.

Na Figura 8(c) a operação de união foi aplicada nos grafos $K_1$ representados em 8(a) e 8(b). Já em 8(f), a operação de junção foi aplicada nos grafos $K_1$ de 8(d) e 8(e). No grafo em 8(g), a operação de junção aplicada nos grafos 8(c) e 8(f).

Com as operações de união e junção é possível definir os cografos recursivamente da seguinte forma:

(i) um vértice - grafo $K_1$ - é um cografo;
(ii) se $G_1, G_2$ são cografos, então $G_1 \cup G_2$ também é;
(iii) se $G_1$ e $G_2$ são cografos, a junção $G_1 * G_2$ também é.

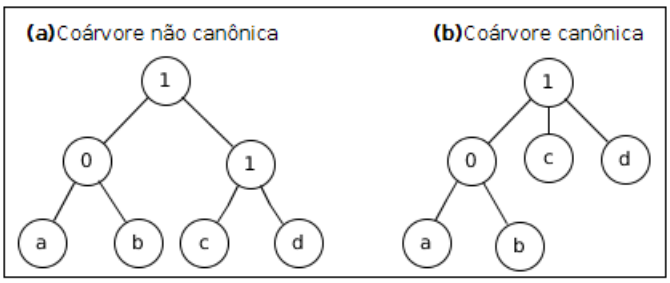
**Figura 8 – Exemplo de geração de cografo**




**Fonte: Autoria própria**

Usando a definição recursiva de construção, cada cografo pode ser associado a uma coárvore. Na coárvore os nós internos representam uma operação de união ou junção e são rotulados com $0$ ou $1$, respectivamente. Além disso, cada nó interno tem pelo menos dois filhos e os nós folhas representam os vértices do grafo. Neste trabalho, dado um cografo $G$, um vértice $v \in V(G)$ e a coárvore $T$ que representa $G$, o mesmo nome, $v$, será usado para referir-se ao vértice $v$ e ao nó que o representa em $T$. Além disso, ao atribuir uma cor ao vértice $v$ da coárvore, esta é também atribuída implicitamente ao vértice $v$ de $G$, ou seja, o nó $v$ de $T$ e o vértice $v$ de $G$ são tratados como o mesmo vértice. Um exemplo de coárvore gerada do grafo resultante da Figura 8(g) é dado na Figura 9(a). Se os níveis da coárvore intercalam as operações de união e junção ela é dita coárvore canônica. A Figura 9(b) apresenta a coárvore canônica do grafo da Figura 8(g). Todo cografo está associado a exatamente uma coárvore canônica (CORNEIL; LERCHS; BURLINGHAM, 1981). Nesse trabalho, consideramos apenas coárvores canônicas, então sempre que é mencionado o conceito de coárvore ela é canônica.

**Figura 9 – Representação duas coárvores do mesmo cografo**




**Fonte: Autoria própria**

Uma propriedade importante da coárvore é baseada no ancestral comum mais próximo de dois vértices $u$ e $v$, denotado por $acp(u, v)$. Um nó $X$ é ancestral de um vértice $y$ - e $y$ é dito seu descendente - se $X$ faz parte do caminho de $y$ até a raiz da árvore. O ancestral comum mais próximo de dois vértices $y$ e $w$ - $acp(y, w)$ - é um nó interno $V$ que é a raiz da menor

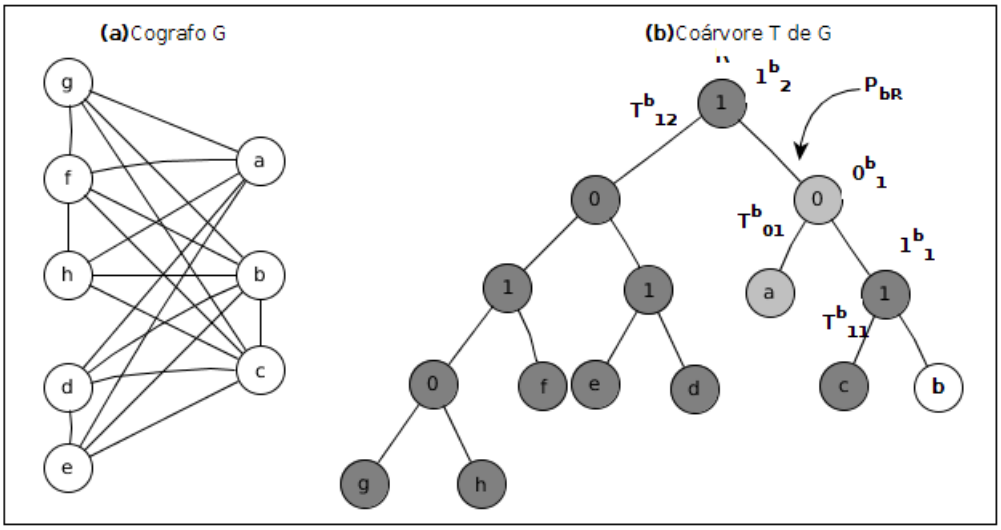
subárvore (subárvore minimal) que contém $y$ e $w$. Na coárvore, dois vértices $y$ e $w$ são adjacentes se $acp(y, w)$ é um nó rotulado com 1. De forma semelhante, dois vértices $y$ e $w$ são não adjacentes se $acp(y, w)$ é um nó rotulado com 0.

Na Figura 9(b), um exemplo de vértices adjacentes são os vértices $a$ e $d$ que tem $acp(a, d) = 1$, e um exemplo de vértices não adjacentes são os vértices $a$ e $b$ que tem $acp(y, w) = 0$. Pode-se conferir que $a$ e $d$ são adjacentes e $a$ e $b$ não são ao se observar a Figura 8(g).

Uma coárvore $T$ é enraizada em $R$ denotada por $T(R)$, onde $R$ é chamado raiz, se $R$ é ancestral comum a todos os demais vértices da árvore. Se $R = 1$ então o grafo é conexo, já se $R = 0$ então o grafo é desconexo. Nesse trabalho, só são considerados grafos conexos, se o grafo for desconexo cada componente deve ser analisada isoladamente e o número biclique cromático é o maior usado na coloração das componentes.

Considere $T(R)$ e $x$ uma folha de $T(R)$, então $P_{xR}$ é o caminho de $x$ a $R$. Como a coárvore é canônica esse caminho, com exceção de $x$, intercala entre nós 0 e 1. Seja $0_i^x$, o i-ésimo nó 0 a partir de $x$ no caminho $P_{xR}$. Da mesma forma, seja $1_i^x$, o i-ésimo nó 1 a partir de $x$ no caminho $P_{xR}$. Cada nó interno em $P_{xR}$ tem um filho neste caminho e no mínimo um filho que não está neste caminho. Na Figura 10 é apresentado um exemplo de caminho $P_{bR}$ do vértice $b$ até a raiz $R$. Note que o nó $0_1^b$, por exemplo, possui o filho $1_1^b$ no caminho e $a$ não pertencente ao caminho.

**Figura 10 – Cografo e sua coárvore com caminho $P_{bR}$**




**Fonte: Autoria própria**

Dessa forma, para cada vértice $y$ de um cografo é possível definir subárvores que são enraizadas em nós internos do caminho $P_{yR}$. Seja $0_i^y$ (resp. $1_i^y$) um nó interno qualquer do caminho. Seja $X$ um filho de $0_i^y$ (resp. $1_i^y$), nó interno ou folha, que pertencente $P_{yR}$. Induza uma subárvore enraizada no nó $0_i^y$ (resp. $1_i^y$) sem $X$ e seus descendentes. Essa subárvore é definida como $T_{0i}^y$ (resp. $T_{1i}^y$). Na Figura 10 no caminho $P_{bR}$ temos as subárvores $T_{11}^b$, $T_{01}^b$ e $T_{12}^b$.

Um subconjunto $M \subseteq V$ é um módulo de $G$ se para todo par de vértices $x, y \in M$, se $v \in V - M$, $(x, v) \in E(G)$ se e somente se $(y, v) \in E(G)$. Para qualquer vértice $y$ de um

cografo quando estabelecido o caminho $P_{yR}$, as folhas de $T^y_{0i}$ (resp. $T^y_{1i}$) definem um módulo, uma vez que quaisquer dois vértices $x$ e $y$ em uma subárvore $T^y_{0i}$ (resp. $T^y_{1i}$) possuem as mesmas adjacências em relação a qualquer vértice $v$ que não está nessa subárvore. Na Figura 10 $\{c\}$ , $\{a\}$ e $\{g, h, f, e, d\}$ são módulos do grafo.

As Propriedades 2.1 e 2.2 relacionam características entre o cografo e a sua coárvore.

**Propriedade 2.1.** *Dado um cografo $G$ e sua coárvore $T$, o grafo $\overline{G}$ é um cografo e sua coárvore $\overline{T}$ é exatamente $T$ com os rótulos $1$ e $0$ invertidos.*

**Propriedade 2.2.** *Dado um cografo $G$ e sua coárvore $T$, qualquer subgrafo induzido $H \subseteq G$ é um cografo e qualquer subárvore induzida enraizada em $t$, onde $t$ é um nó de $T$, é uma coárvore de um subgrafo de $G$.*

A Propriedade 2.2 é uma propriedade hereditária, ou seja, que se aplica a todos os subgrafos induzidos de um grafo.

Ao se estudar problemas computacionais difíceis em classes de grafos específicas, é importante saber se esta classe de grafos pode ser reconhecida em tempo polinomial. Afinal, não é suficiente resolver o problema em tempo polinomial para os grafos da classe se, quando nos deparamos com um grafo qualquer, não conseguirmos responder se ele pertence a esta classe ou não. A próxima seção trata do problema de reconhecimento de cografos.

## 2.1 RECONHECIMENTO DE COGRAFOS

Considere o seguinte problema computacional: dado um grafo $G$, responder "sim"ou "não"para a pergunta: $G$ é um grafo da classe $A$, onde $A$ é um conjunto de grafos pré-definido? Este problema é conhecido como Problema do Reconhecimento de grafos da classe $A$. Por exemplo, sabemos que o conjunto dos cografos é o conjunto dos grafos que não contém um $P_4$ como subgrafo induzido. Sabemos fazer um algoritmo eficiente que reconhece se um grafo qualquer é um cografo? Nessa seção, descreveremos um algoritmo de complexidade linear que Bretscher *et al.* (2008) propuseram para o reconhecimento de cografos baseando nas Propriedades 2.1 e 2.2.

Esse algoritmo de reconhecimento de cografos utiliza um algoritmo de busca em largura especial, conhecido como Busca em Largura Lexicográfica (LexBFS). A LexBFS foi projetada em 1976 para ser utilizada no reconhecimento em tempo linear de grafos cordais[1]. Ela é basicamente uma busca em largura com uma regra adicional. Essa regra consiste em usar como critério de desempate para explorar os vértices, aqueles que possuem como vizinhos vértices que foram visitados mais cedo pela busca.

[1] Um grafo é cordal se todo ciclo de tamanho maior que 3 tem uma corda.

A LexBFS utiliza-se de uma lista $L$ composta de partes $P_i$ e a cada iteração o vértice $v$ mais a esquerda da lista é escolhido como pivô e enumerado. Um vértice é dito visitado quando já foi pivô. No início da busca a partição tem uma única parte e representa a própria lista de vértices de entrada. A lista é particionada em mais conjuntos quando $v$ possui vértices adjacentes e não adjacentes na mesma parte $P_j$. Nesse caso, uma nova parte é criada contendo os vértices adjacentes a $v$ contidos em $P_j$ e posicionada imediatamente à esquerda da parte $P_j$, que contém agora apenas vértices não adjacente a $v$. Com isso, vértices adjacentes ao pivô são visitados mais cedo na busca. A saída do algoritmo é uma sequência $\sigma$ dos vértices na ordem em que eles foram visitados. O Algoritmo 1 apresenta a LexBFS.

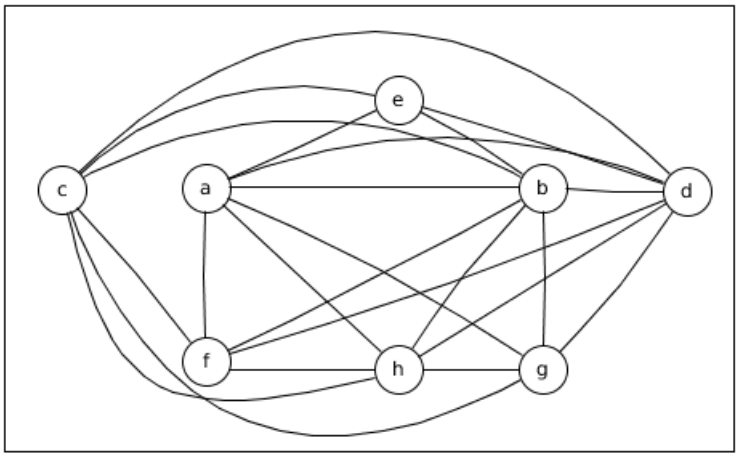
**Algoritmo 1:** LexBFS($G$, $\tau$)

```
   Entrada: Um grafo G = (V, E) e uma ordem τ dos vértices
   Saída: uma ordem σ dos vértices de G
 1 início
 2     L ⟵ τ
 3     σ uma sequência de vértices vazia
 4     enquanto ∃P_i ≠ ∅ em L = (P_1, ..., P_k) (L tiver uma célula não vazia) faça
 5         Seja P_l a célula não vazia mais a esquerda
 6         Remova o primeiro o primeiro vértice, α de P_l
 7         Adicione α ao final de σ
 8         para cada P_j, j ≥ l faça
 9             Seja P' = {v|v ∈ N(x) ∩ P_j}  (P' é a vizinhança aberta de α)
10             se P' é não vazio e P' ≠ P_j então
11                 Remova P' de P_j
12                 Insira P' a esquerda de P_j
13             fim
14         fim
15     fim
16     retorna σ
17 fim
```

**Fonte: Bretscher *et al.* (2008)**

Para exemplo de execução do Algoritmo 1 utilizamos o grafo da Figura 11.

**Figura 11 – Grafo exemplo de geração da coárvore**




**Fonte: Autoria própria**

A permutação de entrada para o algoritmo é $\tau = g, d, a, e, f, h, c, b$. No início há apenas uma célula que contém todos os vértices. Quando a busca é executada pela primeira vez o pivô é o vértice $g$. A parte é particionada em duas células: [d a h c b], cujos vértices são adjacentes a $g$, e [e f], cujos vértices não são adjacentes a $g$. Então, célula [d a h c b] é inserida à direita da célula [e f]. O processo se repete tomando como pivô em cada iteração o primeiro vértice não visitado da sequência. O Quadro 1 mostra todas as iterações do algoritmo no grafo.

**Quadro 1 – Permutação LexBFS**

| Numeração | pivô | Vizinhança não numerada | Células |
|---|---|---|---|
| | | | [g d a e f h c b] |
| 1 | g | { d, a, h, c, b } | [d a h c b] [e f] |
| 2 | d | { a, f, c, b, e, h } | [d h c b] [e f] |
| 3 | a | { f, b, e, h } | [h b] [c] [e f] |
| 4 | h | { b, c, f } | [b] [c] [f] [e] |
| 5 | b | { f, e } | [c] [f] [e] |
| 6 | c | { f, e } | [f] [e] |
| 7 | f | { } | [e] |
| 8 | e | {} | |

**Fonte: Autoria própria**

No final da busca as células são organizadas em fatias. A Definição 2.3 apresenta o conceito de fatia.

**Definição 2.3** (Fatia)**.** *Uma fatia é um conjunto maximal de vértices consecutivos na ordem $\alpha$, de um vértice com numeração $i$ até um vértice com numeração $j$, tal que todos os vértices de $i$ até $j$ possuem a mesma vizinhança no subgrafo induzido pelos vértices com numeração menor ou igual a $i$.*

Cada vértice cria uma fatia. Dado um vértice qualquer $v$, uma fatia que inicia em $v$ é chamada $S(v)$ e é dividida em duas subfatias podendo essas serem vazias. A primeira subfatia denotada por $S^A(v)$ é formada por $S(v) \cap N(v)$, isto é, é uma subfatia que contêm os vértices adjacentes a $v$.

Quando todos os vértices de $S^A(v)$ estão numerados, os vértices de $S(v)$ não adjacentes a $v$ podem ter sido divididos em várias células, as quais formam a segunda subfatia denotada por $S_i(v)$, também chamadas de $v$-células, e representam $S(v) \cap \overline{N}(x) = S_1(v) \cup S_2(v) \cup \cdots \cup S_k(v)$, onde $k$ é o número de células não adjacentes a $v$ contidas na fatia $S(v)$. Por exemplo, quando toda a fatia $S^A(g) =$ [d a h c b] é enumerada, existem duas células não adjacentes em $S(g) \cap \overline{N}(g) =$ [f] [e] onde $S_1(g) = f$ e $S_2(g) = e$. O Quadro 2 apresenta as células $S_i$ para cada vértice do grafo da Figura 11 após a aplicação da LexBFS.

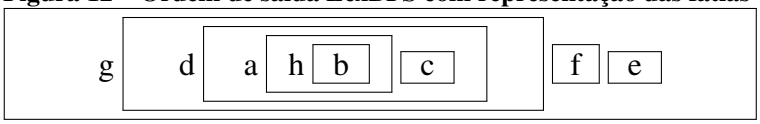
**Quadro 2 – Subfatias não adjacentes $S_i$**

| Vértice | $S_i$(vértice) |
|---|---|
| g | $S_1(g) =$ f $S_2(g) =$ e |
| d | $\emptyset$ |
| a | $S_1(a) =$ c |
| h | $\emptyset$ |
| b | $\emptyset$ |
| c | $\emptyset$ |
| f | $\emptyset$ |
| e | $\emptyset$ |

**Fonte: Autoria própria**

A divisão das fatias na ordem final dos vértices $\sigma$ é apresentada na Figura 12 com a marcação das fatias e subfatias. Pode-se notar, por exemplo, que a fatia $S(a)$ é composta pelos vértices $\{a, h, b, c\}$, onde a subfatia $S^A(a)$ contém o subconjunto os vértices $\{h, b\}$ e a subfatia $S_i(a) = S_1(a) = \{c\}$. Já a fatia $S(d)$ é composta pelo subconjunto de vértices $\{d, a, h, b, c\}$ com subfatia $S^A(d) = \{a, h, b, c\}$ e a subfatia $S_i(d)$ vazia.

**Figura 12 – Ordem de saída LexBFS com representação das fatias**




**Fonte: Autoria própria**

A fatia $S_i$ é utilizada para a construção da coárvore juntamente com a fatia $\overline{S_i}$, que é o conjunto das células não adjacentes no complemento do grafo. Bretscher *et al.* (2008) criaram uma variação para a LexBFS, chamada LexBFS$^-$, que atua sobre o grafo original, mas produz o resultado de uma LexBFS sobre o complemento do grafo. Para isso, a LexBFS$^-$ altera apenas uma linha do algoritmo original, não necessitando realizar a operação de complemento do grafo, o que garante menor custo computacional do algoritmo de reconhecimento. Essa alteração consiste que, na LexBFS$^-$, ao invés de inserir os vizinhos à esquerda da célula a que pertenciam, esses são inseridos à direita. Note que posicionar os vizinhos à direta no grafo original é o mesmo que posicionar os vizinhos à esquerda em uma LexBFS executada no complemento do grafo.

A entrada da LexBFS$^-$, no algoritmo de reconhecimento de cografos, é a ordem de saída $\sigma$ da LexBFS. A permutação resultante da LexBFS$^-$ é aqui representada por $\overline{\sigma}$.

Como exemplo de aplicação da LexBFS$^-$ é utilizado o grafo da Figura 11 recebendo como entrada a sequência $\sigma = g, d, a, h, b, c, f, e$ gerada pela LexBFS, como visto anteriormente. O primeiro vértice selecionado para pivô é $g$. O subconjunto de vértices $\{d, a, h, c, b\}$ são adjacentes a ele no grafo original e o subconjunto $\{e, f\}$ não sã adjacentes. A operação de complemento inverte as adjacências tornando $\{e, f\}$ adjacentes a $g$ e $\{d, a, h, c, b\}$ não adjacentes. Então colocando $\{d, a, h, c, b\}$ à direita estamos inserindo os vizinhos do pivô no complemento do grafo à esquerda. O Quadro 3 mostra todas as iterações do algoritmo no grafo.

**Quadro 3 – Permutação LexBFS$^-$**

| Numeração | pivô | Vizinhança não numerada | Células |
|---|---|---|---|
| | | | g d a h b c f e |
| 1 | g | { d, a, h, c, b } | e f \| d a h c b |
| 2 | e | { d, a, c, b } | f \| h \| d a c b |
| 3 | f | { h, d, a, c, b} | h \| d a c b |
| 4 | h | { d, a, c, b } | d a c b |
| 5 | d | { a, c, b } | a c b |
| 6 | a | { b } | c \| b |
| 7 | c | {} | b |
| 8 | b | {} | |

**Fonte: Autoria própria**

Dado um cografo $G$ com $v \in V(G)$, uma fatia gerada pela LexBFS$^-$ é representada como $\overline{S}(v)$ e as subfatias são representadas como $\overline{S^A}(v)$, para o conjunto de vértices adjacentes a $v$ em $\overline{G}$, e $\overline{S_i}(v) = \overline{S_1}(v) \cup \overline{S_2}(v) \cup \cdots \cup \overline{S_k}(v)$, onde $k$ é o número de células não adjacentes a $v$ em $\overline{G}$ contidas na fatia $\overline{S}(v)$. O Quadro 4 apresenta a subfatia $\overline{S_i}$ para cada vértice do grafo após a aplicação da LexBFS$^-$.

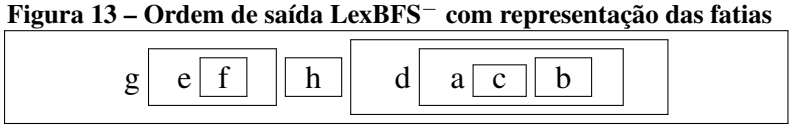
**Quadro 4 – Subfatias não adjacentes $\overline{S_i}$**

| Vértice | $\overline{S_i}$(vértice) |
|---|---|
| g | $\overline{S_1}(g) =$ h $\overline{S_2}(g) =$ d a c b |
| d | $\overline{S_1}(d) =$ a c b |
| a | $\overline{S_1}(a) =$ b |
| h | $\emptyset$ |
| b | $\emptyset$ |
| c | $\emptyset$ |
| f | $\emptyset$ |
| e | $\emptyset$ |

**Fonte: Autoria própria**

A saída do algoritmo $\overline{\sigma} = g, e, f, h, d, a, c, b$ com a marcação das fatias e subfatias é representada na Figura 13.

**Figura 13 – Ordem de saída LexBFS$^-$ com representação das fatias**




**Fonte: Autoria própria**

Após a computação dos dois algoritmos, se o grafo for um cografo, as permutações $\sigma$ e $\overline{\sigma}$ obedecem a Propriedade do Subconjunto da Vizinhança (PSV). Para entender o que é essa propriedade são necessárias as definições que seguem.

**Definição 2.4.** *A vizinhança local de uma célula $S_i(v)$, denotada por $N^l(S_i(v))$, de uma fatia $S(v)$ para uma LexBFS $\sigma$ é o conjunto de vértices em $S(v)$ antes de $v_i$ adjacentes a ao menos um vértice em $S_i(v)$.*

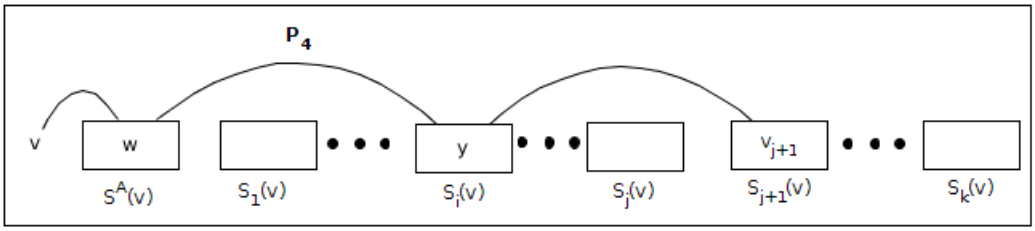
A Definição 2.4 considera que dada uma fatia $S(v)$, a vizinhança local de cada célula não adjacente $S_i(v)$ é composta pelos vértices das células anteriores ao primeiro vértice da célula $S_i(v)$. Esses vértices podem estar tanto em $S^A(v)$ quanto em $S_j(v)$ com $j < i$. Por exemplo, considerando a Figura 12, a fatia $S_i(g)$ contém as células $S_1(g) = f$ e $S_2(g) = e$. A vizinhança local da célula $S_1(g)$ pode estar contida apenas em $S^A(g)$ e a vizinhança local da célula $S_2(g)$ pode estar contida em $S^A(g)$ e $S_1(g)$.

Contudo, para cografos, a Observação 2.5 mostra uma particularidade da classe abordando vizinhança local.

**Observação 2.5.** *Se $\sigma$ é uma LexBFS de um cografo, então $N^l(S_i(v)) = N(S_i(v)) \cap S^A(v)$.*

Ou seja, quando se trata de um cografo, a vizinhança local de uma fatia $S_i(v)$ é composta apenas por vértices na célula $S^A(v)$. Dessa forma, um vértice $x$ de $S_i(v)$ só é adjacente a um vértice $z$ de outra fatia se essa fatia for $S^A(v)$. Isso ocorre pois violando essa observação serão encontrados $P_4$ induzidos no grafo. A Figura 14 ilustra o possível $P_4$ formado quando dois vértices de $v$-células diferentes possuem adjacência.

**Figura 14 – Vizinhança local com $P_4$**




**Fonte: Bretscher *et al.* (2008)**

**Definição 2.6** (Propriedade do Subconjunto da Vizinhança - PSV)**.** *Uma permutação LexBFS obedece à PSV se e somente se*

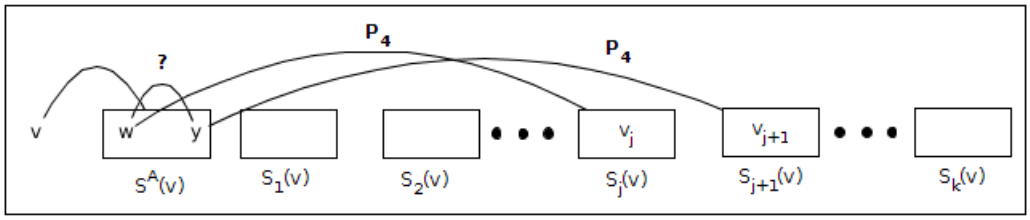
$$\forall v \in V(G), \forall i < j \text{ tais que } S_i(v) \neq \emptyset, \text{ tem-se que } N^l(S_i(v)) \supset N^l(S_j(v))$$

Essa propriedade dita que cada célula $S_i(v)$ possui em sua vizinhança local pelo menos todos os vértices pertencentes às vizinhanças locais das partições de índice maior que $i$. Pela Observação 2.5, a vizinhança local de cada célula está contida em $S^A$ para cografos. A Figura 15 ilustra casos onde a Propriedade do Subconjunto da Vizinhança não é obedecida. Note que há $P_4$ induzido em decorrência da $v$-célula $S_j(v)$ não possuir adjacência com o vértice $y$ enquanto a $v$-célula $S(v_{j+1})$ possui. Note que a aresta $(w, y)$ pode ou não existir. Bretscher *et al.* (2008) apresenta um algoritmo que verifica se a PSV é satisfeita em tempo linear.

A partir da Definição 2.6 temos o Teorema 2.7 que fala sobre a equivalência de cografos e grafos cujas permutações LexBFS e LexBFS$^-$ obedecem à PSV.

**Teorema 2.7.** *Sejam $\sigma$ = LexBFS$(G, \tau)$ para uma permutação $\tau$ qualquer de $V(G)$ e $\overline{\sigma}$ = LexBFS$^-(G, \sigma)$. $G$ é um cografo se e somente se $\sigma$ e $\overline{\sigma}$ obedecem à PSV.*

**Figura 15 – PSV não obedecida**




**Fonte: Bretscher *et al.* (2008)**

Dessa forma, após a aplicação da LexBFS e LexBFS$^-$, se as ordens $\alpha$ e $\overline{\alpha}$ obedecem a $PSV$, o grafo é um cografo e, como consequência, possui uma coárvore canônica. Para a montagem da coárvore, dado um vértice $v$, as subfatias $S_1(v)$ e $\overline{S_1}(v)$ são verificadas e por meio delas o algoritmo pode retornar nós rotulados com $0$ ou $1$ ou ainda intercalar as subfatias $S_i(v)$ e $\overline{S_j}(v)$ para assim gerar vários nós $0$ e $1$ intercalados. Isso se deve ao fato de que cada $S_i(v)$ (resp. $\overline{S_i}(v)$) é um módulo em relação a cada célula da fatia. A notação $v[i]$ indica o primeiro vértice da i-ésima subfatia $S_i(v)$ e $\overline{v}[j]$ o primeiro vértice da $j$-ésima subfatia $\overline{S_j}(v)$.

O algoritmo de construção da coárvore canônica aqui exposto é adaptado de Lima (2014), onde o autor também se baseou no artigo de Bretscher *et al.* (2008). Nessa adaptação separamos os nós internos e nós folhas em listas distintas.

A estrutura dos nós de operação (nós $1$ ou $0$) da coárvore compreende os seguinte campos:

- tipo: pode apresentar valor $0$ ou $1$ dependendo da operação que representa;
- listaFilhosVertices: lista com filhos que representam vértices no cografo;
- listaFilhosOperacao: lista de filhos que representam as operações que ocorreram na geração do grafo.

A variável $T$ é o nó raiz da subárvore que ao final do algoritmo forma a coárvore e $V$ é um nó raiz auxiliar utilizado no retorno das chamadas. O Algoritmo 2, adiciona novos filhos para o nó recebido no primeiro parâmetro, $T$, a partir do nó do segundo parâmetro, $V$. Se o $V$ é vértice, então é adicionado na listaFilhosVertices, se $V$ é operação diferente de $T$, então é inserido na listaFilhosOperacao, entretanto se a operação for a mesma, os filhos de $V$ são inseridos nas listas correspondentes em $T$. Como a coárvore de saída é canônica se a subárvore $V$ tem o mesmo nó de operação na raiz que $T$, o nó raiz de $V$ não pode ser incorporado a $T$, pois resultaria em dois níveis consecutivos de mesmo tipo. O Algoritmo 3 apresenta o algoritmo de construção da coárvore.

**Algoritmo 2:** MESCLARARVORES(NÓ INTERNO $N$, FILHOS $V$)

```
1  início
2      se V é vértice então
3          adicionarFilho(T → listaFilhosVertices, V)
4      fim
5      se V → tipo ≠ T → tipo então
6          adicionarFilho(T → listaFilhosOperacao, V)
7      senão
8          adicionarFilho(T → listaFilhosVertices, V → listaFilhosVertices)
9          adicionarFilho(T → listaFilhosOperacao, V → listaFilhosOperacao)
10     fim
11 fim
```

**Fonte: Autoria própria**

**Algoritmo 3:** COÁRVORE(VÉRTICE $v$)

```
   Entrada: um vértice v de G
   Saída: A coárvore canônica de G
1  início
2      se v[1] = ∅ e v̄[1] = ∅ então
3          retorna {v}
4      fim
5      se v[1] = ∅ então
6          V ← Coárvore(v̄[1])
7          T ← No(1 , v)
8          mesclarArvores(T, V)
9      fim
10     se v̄[1] = ∅ então
11         V ← Coárvore(v[1])
12         T ← No(0 , v)
13         se V é vértice então
14             adicionarFilho(T → listaFilhosVertices, V)
15         fim
16         mesclarArvores(T, V)
17     fim
18     se v[1]v̄[1] ∈ E então
19         w ← 0
20     senão
21         w ← 1
22     fim
23     T ← v; k ← 1; i ← 1
24     enquanto (w = 0 e v[i] ≠null)ou(w = 1 e v̄[i] ≠null) faça
25         se w = 0 então
26             u ← v[i]
27         senão
28             u ← v̄[i]
29         fim
30         V ← Coárvore(u)
31         T ← No(w , T)
32         mesclarArvores(T, V)
33         w ← 1 − w; k ← k + 1; i ← ⌈k/2⌉
34     fim
35     retorna T
36 fim
```

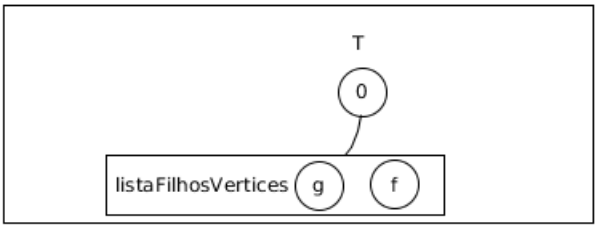
**Fonte: Adaptado de Lima (2014)**

Na criação de um nó, o primeiro parâmetro é o tipo do nó, e o segundo a subárvore construída até o momento. Fica subentendido que é necessário atribuir à lista correta, listaFilhosVertices ou listaFilhosNos, a raiz da subárvore já existente para o novo nó que é tornado nova raiz da subárvore.

Vamos aplicar o Algoritmo 3 no grafo da Figura 11 para exemplo de execução do algoritmo. Temos a sequência $\alpha$ e $\overline{\alpha}$ assim como as subfatias $S_i$ e $\overline{S_i}$ já calculadas anteriormente. O primeiro vértice das sequências $\alpha$ e $\overline{\alpha}$ é $g$, então ele será o vértice de entrada na primeira execução do algoritmo.

A subfatia, $S_i(g) = S_1(g) =$ f $\cup\ S_2(g) =$ e, e a subfatia $\overline{S_i}(g) = \overline{S_1}(g) =$ h $\cup\ \overline{S_2}(g) =$ d a c b. Como as duas subfatias não são vazias é verificado se $(g[1], \overline{g}[2])$ pertence ao conjunto de arestas, ou seja, se $(f, h)$ é uma aresta do grafo. Como $(f, h)$ é uma aresta do grafo, a variável $w$ recebe valor $0$, onde $w$ que é o tipo do nó interno que é criado cada iteração do *while* da linha 24. Como $w$ possui valor $0$, a variável $u$ recebe $g[1]$, que é o vértice $f$. Em seguida, $V$ recebe o retorno da chamada do algoritmo enviando $u$ como parâmetro. Dentro dessa chamada, como $u$ tem o valor $f$, e já que esse vértice tem $S_i$ e $\overline{S_i}$ vazios, o próprio $f$ é retornado. Ao retornar dessa chamada, é criado um nó $0$ na linha 31, que se torna a nova raiz, com $g$ na listaFilhosVertices. Ao mesclar $T$ e $V$, $f$ é adicionado na listaFilhosVertices. A Figura 16 ilustra como está a coárvore nesse momento, as ilustrações referentes a explicação desse algoritmo não seguem criteriosamente as ligações entre nós criadas pelo algoritmo, mostram apenas uma forma de visualização para auxiliar o leitor no entendimento mantendo o padrão das imagens de coárvore já apresentadas.

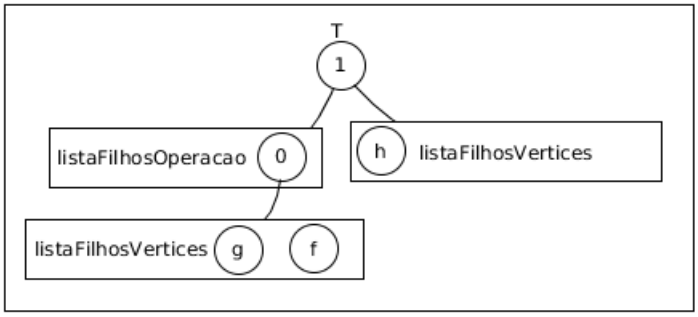
**Figura 16 – Exemplo de geração da coárvore passo 1**




**Fonte: Autoria própria**

Nesse ponto na linha 33, $w$ tem seu valor invertido, como era $0$ passa a ser $1$, e $i$ é incrementado de forma que a cada duas iterações do *while* seu valor aumente em um. Agora que $w$ tem valor $1$ a subfatia $\overline{S_i}(g)$ é analisada. Como $i$ continua com valor $1$, $u$ recebe $h$ que é $\overline{g}[1]$. Coárvore($u$) atribui o próprio $h$ para $V$, então é criado um nó $1$ que tem a subárvore já construída, $T$, como filho. Depois, $V$ e $T$ são mescladas. A ideia sobre como está a coárvore nesse momento é mostrada na Figura 17.

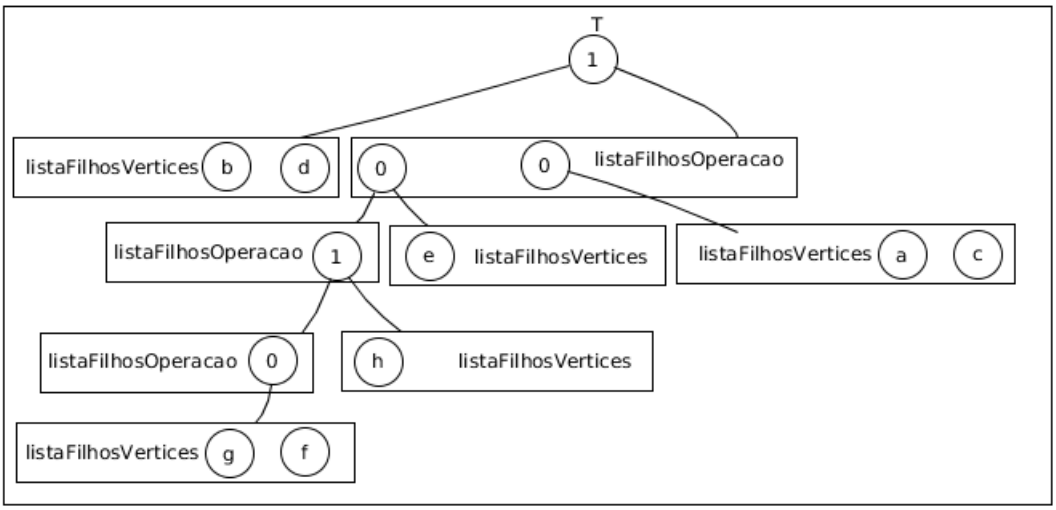
**Figura 17 – Exemplo de geração da coárvore passo 2**




**Fonte: Autoria própria**

O resultado ao final de todas as chamadas é apresentado a coárvore da Figura 18.

**Figura 18 – Exemplo de geração da coárvore final**




**Fonte: Autoria própria**

O Algoritmo 4 apresenta todos os passos para o reconhecimento de cografos.

**Algoritmo 4:** RECONHECIMENTO($G$)

**Entrada:** Um grafo $G = (V, E)$
**Saída:** uma coárvore $T$ se $G$ for um cografo ou $null$ caso contrário

```
1  início
2      Seja τ uma ordem qualquer de V(G)
3      σ ← LexBFS (G, τ)
4      σ̄ ← LexBFS⁻ (G, σ)
5      se σ e σ̄ obedecem à PSV então
6          retorna Construção_Coárvore(v, M);
7      senão
8          retorna null
9      fim
10 fim
```

**Fonte: Bretscher *et al.* (2008)**

Bretscher *et al.* (2008) provaram que a execução do Algoritmo 4 tem complexidade de tempo $O(|V(G)| + |E(G)|)$, onde $|V(G)|$ é o número de vértices e $|E(G)|$ é o número de

arestas. O algoritmo recebe uma sequência de vértices como entrada e executa inicialmente a LexBFS sobre uma ordem qualquer $\tau$ dos vértices produzindo como saída uma ordem $\alpha$ dos vértices. A ordem $\alpha$ é entrada para a segunda etapa do algoritmo, a LexBFS$^-$, que tem como saída a ordem $\overline{\alpha}$. Sobre as saídas $\alpha$ e $\overline{\alpha}$ é realizada a verificação da Propriedade do Subconjunto da Vizinhança (PSV). Se $\alpha$ e $\overline{\alpha}$ obedecerem a PSV o algoritmo de construção da coárvore é chamado, caso contrário, é retornado, nessa adaptação, *null*. No algoritmo original, é retornado um $P_4$ induzido.

A partir das características da coárvore é possível fazer a contagem de cliques de forma eficiente (CORNEIL; LERCHS; BURLINGHAM, 1981). No próximo capítulo é abordado o Problema de Enumeração de Bicliques, onde a coárvore é utilizada para enumerar e contar os conjuntos independentes maximais, no complemento do grafo as cliques maximais, e a partir deles contar e enumerar as bicliques.

# 3 NOVOS RESULTADOS EM ENUMERAÇÃO DE BICLIQUES EM COGRAFOS

Nesse capítulo discorremos sobre como se estruturam as bicliques na coárvore e apresentamos um algoritmo que enumera todas as bicliques de um dado cografo. Para tanto, a Seção 3.1 apresenta propriedades da coárvore quanto aos conjuntos independentes maximais, assim como a contagem e enumeração desses conjuntos. A partir disso, a Seção 3.2 mostra como contar e enumerar as bicliques de um cografo.

## 3.1 CONJUNTOS INDEPENDENTES MAXIMAIS NA COÁRVORE

Para encontrar conjuntos independentes maximais na coárvore, nos aproveitamos do rotulamento dos nós internos que nos dizem se dois vértices possuem ou não adjacência por meio do ancestral comum mais próximo. Os Lemas 3.1 e 3.2 definem como se estruturam os conjuntos independentes maximais na coárvore.

**Lema 3.1.** *Dado um cografo $G$ e sua coárvore $T$. Seja $x$ um nó rotulado com $1$ em $T$ e $S$ um conjunto independente maximal em $G$ contido nos descendentes de $x$. Então todo elemento de $S$ é descendente de uma mesma subárvore enraizada em um dos filhos de $x$.*

*Demonstração.* Sejam $a$ e $b$ dois filhos quaisquer de $x$. Como $x$ é um nó de junção existe aresta de cada vértice da subárvore $T(a)$ com todos os vértices da subárvore $T(b)$. Dessa forma, um conjunto independente pode conter descendentes de apenas uma dessas subárvores. □

**Lema 3.2.** *Seja um cografo $G$, sua coárvore $T$ e $T(x)$ uma subárvore enraizada em um nó $x$ rotulado com $0$. Qualquer conjunto independente maximal no subgrafo induzido pelos descendentes de $T(x)$ contém pelo menos um vértice de cada subárvore enraizada em um filho de $x$.*

*Demonstração.* Considere um cografo $G$, sua coárvore $T$ e $T(x)$ uma subárvore enraizada em um nó $x$ rotulado com $0$. Considere um conjunto independente maximal $S$ no subgrafo induzido pelos descendentes de $T(x)$. Suponha, por contradição, que existe um nó $y$, filho de $x$, tal que nenhum vértice em $T(y)$ pertence a $S$. Como a operação de união que ocorre no nó $x$ não cria arestas, os vértices em $T(y)$ não são adjacentes aos vértices que descendem de outras subárvores enraizadas em filhos de $x$. Seja $v$ um vértice de $G$ que pertence a $T(y)$. Então, $S \cup \{v\}$ é um conjunto independente, contradizendo a maximalidade de $S$. □

**Observação 3.3.** *Considere um cografo $G$, sua coárvore $T$ e uma subárvore $T(x)$ enraizada em um nó $0$, como no Lema 3.2. Se uma subárvore enraizada em um filho de $x$ se constitui de um único vértice $v$ do grafo $G$, então $v$ pertence a todos os conjuntos independentes maximais que contêm descendentes de $x$.*

A partir dos Lemas 3.1 e 3.2 e da Observação 3.3, é possível contar os conjuntos independentes maximais de um cografo em tempo linear. Denotamos o número de conjuntos independentes maximais do subgrafo induzido pelos vértices descendentes de um nó $v$ da coárvore como $i(v)$. Dizemos que um subconjunto de vértices $A$ é enraizado em um nó $0$ (resp. $1$) se a menor subárvore que contém todos os vértices de $A$ está enraizado em um nó $0$ (resp. $1$).

Considere um cografo $G$ e sua coárvore $T$. Suponha que deseja-se saber quantos conjuntos independentes maximais existem no subgrafo de $G$ induzido pelos vértices que pertencem à $T(v)$. Se $T(v)$ tem apenas um nó, então $v$ é uma folha e é um conjunto independente maximal. Neste caso, $i(v) = 1$.

Se $v$ é um nó com rótulo $0$, pelo Lema 3.2 toda subárvore enraizada nos filhos de $v$ terá pelo menos um descendente em qualquer conjunto independente maximal do subgrafo induzido pelos descendentes de $T(v)$. Dessa forma, o número de conjuntos independentes maximais será resultado de todas as combinações possíveis entre os conjuntos independentes maximais de cada subárvore enraizada em um filho de $v$. Isso resulta que a contagem dos conjuntos independentes em nós $0$'s é a multiplicação do número de conjuntos independentes dos seus filhos. Ou seja, $i(v) = \Pi_{i=1}^{k} i(v_i)$, onde $\{v_1, v_2, \ldots, v_k\}$ é o conjunto de todos os filhos do nó $v$ na coárvore $T(v)$.

Se $v$ é um nó com rótulo $1$, pelo Lema 3.1, qualquer conjunto independente enraizado em $v$ descende de um único filho de $v$. Então o número total de conjuntos independentes é a soma dos conjuntos independentes maximais em cada filho do nó $1$. Ou seja, $i(v) = \Sigma_{i=1}^{k} i(v_i)$, onde $\{v_1, v_2, \ldots, v_k\}$ é o conjunto de todos os filhos do nó $v$ na coárvore $T(v)$. O Algoritmo 5 faz, recursivamente, a contagem do número de conjuntos independentes maximais em uma coárvore enraizada em um nó $v$.

**Algoritmo 5:** ContagemDeConjuntosIndependentes(nó $v$)

```
Entrada: nó v, na primeira chamada é a raiz da coárvore
Saída: número de conjuntos independentes maximais da subárvore enraizada em v
1  início
2      se v é vértice (nó folha) então
3          retorna 1
4      fim
5      se v é nó rotulado com 0 então
6          i(v) = 1
7          para cada filho v_i com i variando de 1 a k onde k é o número de filhos de v faça
8              i(v) = i(v)* ContagemDeConjuntosIndependentes(v_i)
9          fim
10         retorna i(v)
11     fim
12     se v é nó rotulado com 1 então
13         i(v) = 0
14         para cada filho v_i com i variando de 1 a k onde k é o número de filhos de v faça
15             i(v) = i(v)+ ContagemDeConjuntosIndependentes(v_i)
16         fim
17         retorna i(v)
18     fim
19 fim
```

**Fonte: Autoria própria**

A complexidade do Algoritmo 5 é linear. Observe que a coárvore tem $n$ folhas e no máximo $n-1$ nós internos. Cada chamada do Algoritmo 5 para uma folha, executa apenas a comparação da linha 2. Cada chamada do Algoritmo 5 para um nó interno, executa no máximo as três comparações das linhas 2, 5 e 12, além de uma atribuição (na linha 6 ou 13) e um certo número de execuções das $c$ operações do laço da linha 7 ou da linha $14$, onde $c$ é uma constante positiva. Considerando o total de todas as chamadas recursivas, a soma do número de execuções das linhas 7 e 14 é limitado pelo número de arestas da coárvore. O número de arestas na coárvore é menor que o número de nós (vértices e nós internos), que é $2n-1$. Então, o número de instruções executadas é limitado por $n+(n-1)3+(2n-1)c \leq (2c+4)n$. Portanto, o número de instruções executadas pelo Algoritmo 5 é $O(n)$. Logo, o Algoritmo 5 é linear.

Com um algoritmo semelhante ao de contagem podemos enumerar os conjuntos independentes maximais do cografo. Para tanto, quando o Algoritmo 6 é chamado tendo como parâmetro um nó $v$, e retorna uma lista com todos os conjuntos independentes maximais da coárvore $T(v)$. Se $v$ é um nó $0$, então seus conjuntos independentes maximais são todas as possíveis combinações com um conjunto independente maximal de cada subárvore enraizada em um filho de $v$, então as listas de seus filhos devem ser combinadas de forma que cada combinação de conjuntos independentes de filhos distintos forme um conjunto independente do nó $0$. Se $v$ é um nó $1$, então seus conjuntos independentes maximais são os mesmos retornados por cada um de seus filhos, assim as listas de seus filhos são retornados em uma lista ao pai do nó $1$. No Algoritmo **??**, $L(v)$ representa a lista de conjuntos independentes maximais da coárvore $T(v)$ e $A(v)$ é uma lista auxiliar para nós $0$. O procedimento Combinação(A,k), retorna todas as listas resultantes da combinação de um elemento de cada $A(i)$ com $1 \leq i \leq k$.

**Algoritmo 6:** EnumeracaoDeConjuntosIndependentes(nó $v$)

```
   Entrada: nó v, na primeira chamada é a raiz da coárvore
   Saída: lista de conjuntos independentes da subárvore enraizada em v
 1 início
 2     se v é vértice (nó folha) então
 3         retorna v
 4     fim
 5     se v é nó rotulado com 0 então
 6         A(v) = vazio
 7         para cada filho v_i com i variando de 1 a k onde k é o número de filhos de v faça
 8             A(i) = EnumeracaoDeConjuntosIndependentes(v_i)
 9         fim
10         L(v) = Combinação(A, k)
11         retorna L(v)
12     fim
13     se v é nó rotulado com 1 então
14         L(v) = vazio
15         para cada filho v_i com i variando de 1 a k onde k é o número de filhos de v faça
16             L(v) = L(v)+ EnumeracaoDeConjuntosIndependentes(v_i)
17         fim
18         retorna L(v)
19     fim
20 fim
```

**Fonte: Autoria própria**

Observe que o Algoritmo 6 seria linear, como o Algoritmo 5, se não fosse pela linha $10$, onde o procedimento Combinação(A,k) cria listas com todas as combinações possíveis com $k$ conjuntos independentes maximais, um de cada filho de $v$. Esse número de combinações é o produto $|A(1)| \times |A(2)| \times \ldots \times |A(k)|$. Este número pode ser fatorial em relação ao tamanho da entrada. Ressalta-se que o problema de enumerar todos os conjuntos independentes maximais com tamanho fixo em um grafo qualquer é um problema NP-difícil (LAWLER; LENSTRA; KAN, 1980).

O Algoritmo 6 pode ser utilizado para a enumeração de cliques maximais de um grafo. Para isso, é necessário inverter o rótulo dos nós de operação como enunciado pela Propriedade 2.1. Dessa forma, teremos os conjuntos independentes maximais no complemento do grafo que é o mesmo que as cliques maximais no grafo original.

## 3.2 BICLIQUES NA COÁRVORE

Sabendo como encontrar os conjuntos independentes na coárvore pode-se encontrar as bicliques maximais no cografo. Isso é possível pois uma biclique é formada por dois conjuntos independentes maximais que possuem todas as adjacências entre si e a operação de junção gera as adjacências entre conjuntos independentes. Nota-se, então, que toda biclique é enraizada em um nó com rótulo $1$, como está enunciado no Lema 3.4.

**Lema 3.4.** *Seja $G$ um cografo com coárvore canônica $T$, então toda biclique $Q$ de $G$ é um subconjunto de vértices contidos em uma subárvore minimal $T(x)$, onde $x$ é rotulado com $1$, e os conjuntos independentes maximais $A$ e $B$ de $Q$ são um subconjunto de vértices de exatamente dois filhos de $T(x)$, $u_1$ e $u_2$, tais que $A$ é um subconjunto dos descendentes de $u_1$ e $B$ é um subconjunto dos descendentes de $u_2$.*

*Demonstração.* Primeiro provaremos que uma biclique é formada pelo subconjunto de vértices de uma subárvore minimal enraizada em um nó $1$. Como $Q$ é uma biclique formada pelos conjuntos independentes $A$ e $B$, houve, durante a construção de $G$, uma operação de junção que criou as arestas entre vértices de $A$ e $B$. Logo, um nó $x$ com rótulo $1$ é o ancestral comum mais próximo dos vértices de $A$ e $B$, ou seja, $T(x)$ é a menor árvore que contém todos os vértices da biclique.

Pela Propriedade 3.1 um conjunto independente enraizado em nó $1$ tem descendentes em apenas uma subárvore enraizada em um filho $y$ qualquer de $x$. Dessa forma, para formar uma biclique são necessários descendentes de dois filhos $u_1$ e $u_2$ de $x$.

□

Para contar o número de bicliques em um cografo, pelo Lema 3.4, é suficiente contar quantas bicliques existem na subárvore enraizada em cada nó $1$ da coárvore canônica. Para saber

quantas bicliques existem em um nó rotulado com 1 da coárvore é necessário contar quantas combinações de conjuntos independentes de duas subárvores filhas distintas são possíveis nesse nó. Assim, dado um cografo $G$ e sua coárvore $T$. O número de bicliques maximais de $G$ em um nó $v$ rotulado com 1 é $\sum_{j=1}^{k-1} \sum_{i=j+1}^{k} i(v_j) * i(v_1)$, onde $k$ é o número de filhos de $v$.

O Algoritmo 7 imprime todas as bicliques do grafo utilizando a coárvore. Para isso, o Algoritmo 6 é adaptado imprimindo combinações de conjuntos independentes dois a dois quando o $v$ é um nó 1.

**Algoritmo 7:** ENUMERACAODEBICLIQUES(NÓ $v$)

```
Entrada: nó v, na primeira chamada é a raiz da coárvore
Saída: lista de bicliques da subárvore enraizada em v
1  início
2      se v é vértice (nó folha) então
3          retorna v
4      fim
5      se v é nó rotulado com 0 então
6          A(v) = vazio
7          para cada filho v_i com i variando de 1 a k onde k é o número de filhos de v faça
8              A(v)[i] = EnumeracaoDeBicliques(v_i)
9          fim
10         enquanto houver combinações diferentes faça
11             L(v) = combinação com um conjunto independente de cada posição de A(v)
12         fim
13         retorna L(v)
14     fim
15     se v é nó rotulado com 1 então
16         L(v) = vazio
17         A(v) = vazio
18         para cada filho v_i com i variando de 1 a k onde k é o número de filhos de v faça
19             A(v)[i] = EnumeracaoDeBicliques(v_i)
20             L(v) = L(v) + A(v)[i]
21         fim
22         enquanto houver combinações diferentes faça
23             imprima combinação de dois conjuntos independentes um de cada A(v)[i]
24         fim
25         retorna L(v)
26     fim
27 fim
```

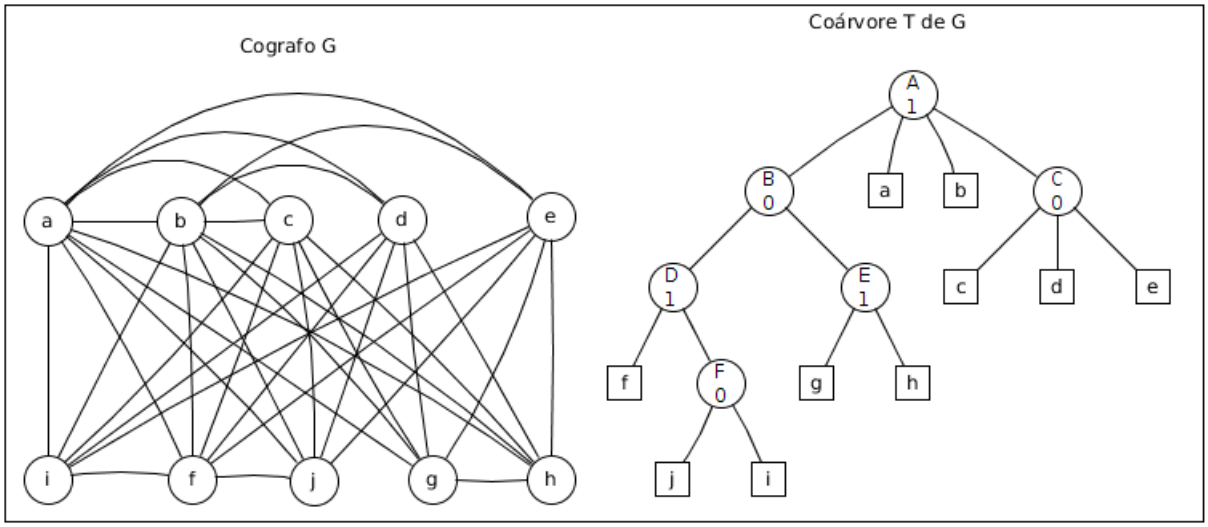
**Fonte: Autoria própria**

Realizando a aplicação do Algoritmo 7 na coárvore $T$ do grafo $G$ apresentado na Figura 19, são geradas 23 bicliques listadas abaixo:

- $\{\{f\},\{j,i\}\}$ ;
- $\{\{g\},\{h\}\}$;
- $\{\{a\}\{b\}\}$ ;
- $\{\{a\}\{f,g\}\}$ ;
- $\{\{a\}\{f,h\}\}$;
- $\{\{a\}\{j,i,g\}\}$;
- $\{\{a\}\{j,i,h\}\}$;

- $\{\{a\}\{c,d,e\}\}$;
- $\{\{b\}\{f,g\}\}$;
- $\{\{b\}\{f,h\}\}$;
- $\{\{b\}\{j,i,g\}\}$;
- $\{\{b\}\{j,i,h\}\}$;
- $\{\{b\}\{c,d,e\}\}$;
- $\{\{f,g\}\{f,h\}\}$;
- $\{\{f,g\}\{j,i,g\}\}$;
- $\{\{f,g\}\{j,i,h\}\}$;
- $\{\{f,g\}\{c,d,e\}\}$;
- $\{\{f,h\}\{j,i,g\}\}$;
- $\{\{f,h\}\{j,i,h\}\}$;
- $\{\{f,h\}\{c,d,e\}\}$;
- $\{\{j,i,g\}\{j,i,h\}\}$;
- $\{\{j,i,g\}\{c,d,e\}\}$;
- $\{\{j,i,h\}\ \{c,d,e\}\}$.

**Figura 19 – Encontrando Bicliques em Cografos**




**Fonte: Autoria própria**

No próximo capítulo apresentamos propriedades da coloração biclique e soluções existentes para o Problema da Coloração Biclique em subclasses dos cografos.

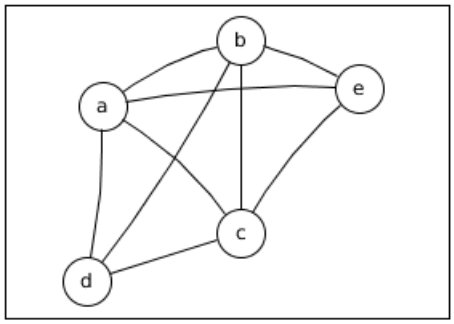
# 4 RESULTADOS CONHECIDOS EM COLORAÇÃO BICLIQUE

Nesse capítulo discorremos sobre as propriedades da coloração biclique e as subclasses dos cografos para as quais se conhece solução polinomial para o Problema da Coloração Biclique.

## 4.1 PROPRIEDADES DA COLORAÇÃO BICLIQUE

Primeiro vamos definir formalmente alguns conceitos necessários para entendimento dessa seção. A vizinhança aberta de um dado vértice $v$ é o conjunto denotado por $N(v)$ formado pelos vizinhos de $v$. Na Figura 20 $N(d) = \{a, b, c\}$. A vizinhança fechada de $v$ é o subconjunto $N[v] = N(v) \cup v$. Na Figura 20 $N[d] = \{a, b, c, d\}$. Dois vértices $v$ e $w$ são ditos gêmeos fracos se $N(w) = N(v)$ e gêmeos fortes se $N[w] = N[v]$. Na Figura 20 $N(d) = N(e) = \{a, b, c\}$, então $d$ e $e$ são gêmeos fracos e $N[a] = N[b] = N[c] = \{a, b, c, d, e\}$ então $a, b$ e $c$ são gêmeos fortes. Um bloco é um subconjunto maximal de vértices gêmeos fortes, isto é, um subconjunto de gêmeos fortes que não está contido em outro. Note que na Figura 20 $\{a, b, c\}$, $\{d\}$ e $\{e\}$ são os blocos maximais do grafo.

**Figura 20 – Conceitos de Vizinhança**




**Fonte: Autoria própria**

**Definição 4.1.** *(TERLISKY, 2010) O número de bloco de $G$, denotado por $\beta(G)$, é o tamanho do maior bloco de $G$.*

No grafo da Figura 20 $\beta(G) = 3$. É importante ressaltar que clique máxima não deve ser confundida com bloco máximo, note que no grafo da Figura 20 as clique máximas têm tamanho $4$, uma formada pelo subconjunto de vértices $\{a, b, c, d\}$ e outra por $\{a, b, c, e\}$, enquanto o maior bloco tem tamanho $3$, formado pelos vértices $\{a, b, c\}$.

**Lema 4.2.** *(TERLISKY, 2010) Uma biclique contém apenas a aresta $(v, w)$ se e somente se $v$ e $w$ são gêmeos fortes.*

*Demonstração.* Suponha que $v$ e $w$ não são gêmeos fortes, isto é, existe um vértice $z$ adjacente a um deles e não ao outro. Sem perda de generalidade, suponha que $z$ é adjacente a $v$ e não a $w$. Então a biclique $\{\{v\},\{w\}\}$ está contida em $\{\{v\},\{w,z\}\}$. Consequentemente, $\{\{v\},\{w\}\}$ não é uma biclique maximal.

Agora, suponha que $\{\{v\},\{w\}\}$ não é uma biclique maximal, então está contida em uma biclique maximal $B$. Logo, existe um vértice $z \in B$. Sem perda de generalidade suponha que o bipartido completo $\{\{v\},\{w,z\}\}$ pertence a $G$. Então, $z$ é adjacente a $v$ e não é adjacente a $w$ □

Por consequência do Lema 4.2 temos a seguinte propriedade:

**Propriedade 4.3.** *(TERLISKY, 2010) Para todo grafo G,* $\chi_{bc}(G) \geq \beta(G)$.

Um limite superior para a coloração biclique de $G$ é $\chi(G)$, número cromático de $G$, o menor número de cores necessário para uma coloração própria de vértices do grafo $G$. Uma coloração de vértices é dita própria se para quaisquer dois vértices $x$ e $y$ adjacentes, as cores de $x$ e $y$ são distintas. Cografos são uma subclasse de grafos perfeitos e para esses o Problema da Coloração Própria de Vértices está resolvido (GOLUMBIC, 2004). Grafos perfeitos admitem uma coloração própria de vértices com $\chi(G) = \omega(G)$, onde $\omega(G)$ é o tamanho da maior clique. Como qualquer par de vértices adjacentes terá cor diferente em uma coloração própria, toda biclique é policromática. A Propriedade 4.4 relaciona o número de cores necessárias para uma coloração biclique com o número cromático.

**Propriedade 4.4.** *(TERLISKY, 2010) Para todo grafo G,* $\chi_{bc}(G) \leq \chi(G)$.

**Propriedade 4.5.** *(TERLISKY, 2010) Se* $v$ *e* $w$ *são dois vértices gêmeos fracos, então toda biclique maximal que contem* $v$ *também contém* $w$.

*Demonstração.* Considere uma biclique maximal $\{X, Y\}$. Vamos mostrar que se $v \in X$, então, a biclique também contém $w$. Se $v \in X$ em uma biclique maximal $\{X, Y\}$, então $Y \subseteq N(v)$ e $X \cap N(v) = \emptyset$. Como $v$ e $w$ são gêmeos fracos, $N(v) = N(w)$ por definição. Então, $Y \subseteq N(w)$ e $X \cap N(w) = \emptyset$. Pela definição de bipartido completo, $\{X \cup \{w\}, Y\}$ é bipartido completo. Portanto, $w$ pertence à biclique maximal $\{X, Y\}$.

□

## 4.2 COLORAÇÃO BICLIQUE EM COGRAFOS

Nessa seção iremos apresentar como o Problema da Coloração Biclique foi resolvido para duas subclasses dos cografos: grafos completos e grafos *threshold*.

### 4.2.1 Grafos Completos

Um grafo é dito completo, denotado por $K_n$, se cada vértice $v \in V(K_n)$ é adjacente a qualquer $y \in V(K_n)$ com $v \neq y$.

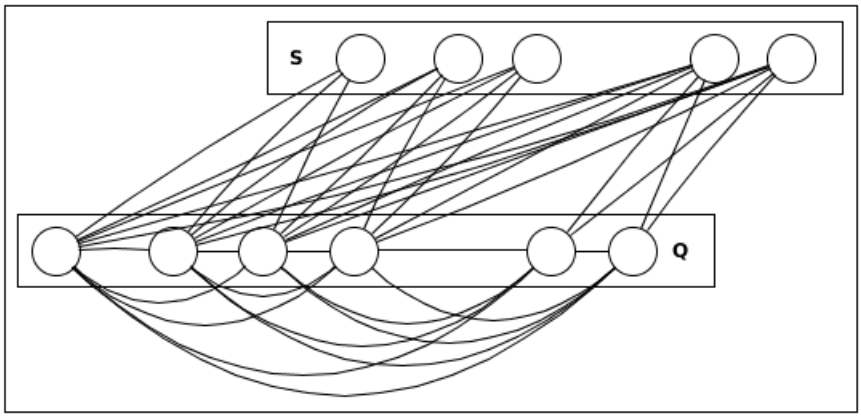
**Corolário 4.6.** *(TERLISKY, 2010) Seja $G$ um grafo com $n$ vértices. Então, $\chi_{bc}(G) = n$ se e somente se $G = K_n$.*

*Demonstração.* Para ver que $\chi_{bc}(K_n) = n$, note que $\beta(K_n) = n$ e pela Propriedade 4.3, $\chi_{bc}(K_n) \geq \beta(K_n)$ . Por outro lado, quando um grafo não é completo, pode-se colorir seus vértices com menos que $n$ cores de forma que vértices vizinhos não tenham a mesma cor. Para ver isso, considere um grafo $G$ com $n$ vértices que não é completo. Então, existem dois vértices $u$ e $v$ em $G$ que não são vizinhos. Pinte os vértices $u$ e $v$ com a mesma cor e use uma cor diferente para cada um dos outros $n-2$ vértices. Então $\chi(G) \leq n-1$. Pela Propriedade 4.4, $\chi_{bc}(G) \leq \chi(G) < n$. □

### 4.2.2 Grafos *Threshold*

Um grafo é *threshold* se é um cografo e seu conjunto de vértices pode ser particionado em dois subconjuntos de forma que um deles é uma clique $Q$ e o outro um conjunto independente $S$. Um exemplo de um grafo *threshold* está na Figura 21. O Problema da Coloração Biclique para os *threshold* foi resolvido por Terlisky (2010). Nessa subseção é dada uma noção de como o problema foi resolvido nessa subclasse dos cografos.

**Figura 21 – *Threshold***




**Fonte: Autoria própria**

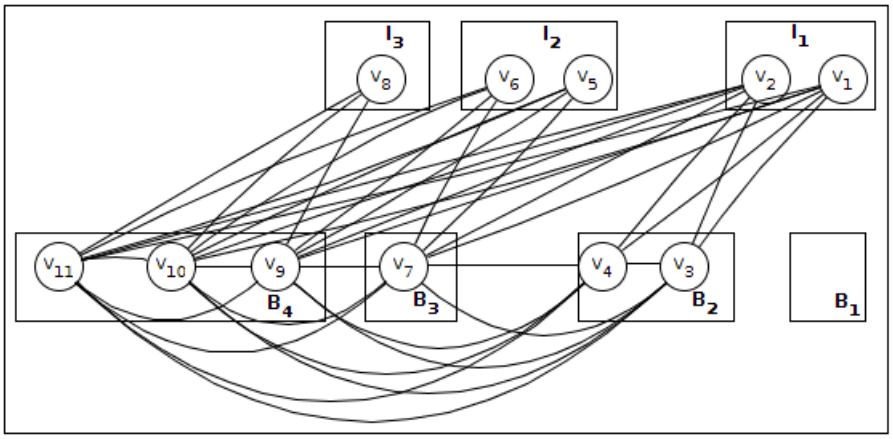
Um vértice $x$ é dito isolado em um conjunto de vértices $X$ se não é adjacente a qualquer outro vértice em $X$ e é dito universal se possui adjacência com todos os vértices de $X$.

Uma ordenação *threshold* é uma sequência de vértices $v_1 v_2 v_3 \ldots v_n$ de forma que $v_i$ é um vértice isolado ou universal no conjunto $X_i = \{v_j : 1 \leq j < i\}$, para qualquer inteiro

$i \in [1, n]$. Note que um vértice $v_i$ é isolado se e somente se faz parte do conjunto independente $S$.

Uma sequência de conjuntos $B_1, I_1, B_2, I_2, \ldots, B_r, I_r$ de $G$ é uma sequência que agrupa todos os vértices do mesmo tipo na ordenação *threshold*, onde $B_1$ contém todos os vértices consecutivos do tipo universal até o primeiro isolado (não incluso) e $I_1$ os vértices consecutivos do tipo isolado até o primeiro do tipo universal (não incluso) e assim sucessivamente até $B_r$. Consideramos apenas grafos conexos, então $I_r$ sempre será vazio. Como $v_1$ e $v_2$ podem ser do tipo isolado em alguns casos, $B_1$ também pode ser vazio, os outros conjuntos não podem ser vazios. A Figura 22 ilustra um exemplo de conjuntos de uma ordenação *threshold*.

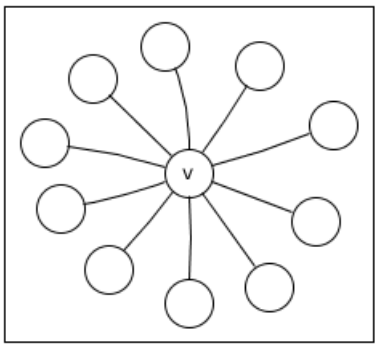
**Figura 22 – Sequência dos conjuntos de um *Threshold***




**Fonte: Autoria própria**

Um $C_n$ é um caminho fechado com $n$ vértices, ou seja, um caminho onde $v_1 v_n \in E(C_n)$. Se um grafo é *Threshold*, então não contém $C_4$ como subgrafo induzido. Então as únicas bicliques possíveis são estrelas. Uma biclique estrela é um subgrafo bipartido completo $K_{1,|Y|} = \{\{v\}, \{Y\}\}$, onde o vértice $v$ é o centro e os vértices de $Y$ são adjacentes apenas a $v$. Um exemplo de estrela é apresentado na Figura 23.

**Figura 23 – Estrela**

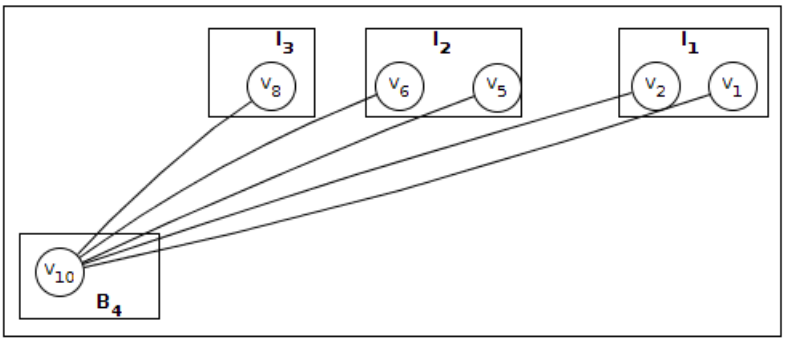



**Fonte: Autoria própria**

**Observação 4.7.** *(TERLISKY, 2010) Seja $G$ um grafo threshold, com ordenação threshold $B_1, I_1, B_2, I_2, \ldots, B_r$ e $D$ uma biclique maximal de $G$. Então algum desses três casos é satisfeito:*

1. *$D$ é uma estrela com centro em $v \in B_i$, e pontas em $I_j$ com $j < i$. Esse caso só é possível se $B_1 = \emptyset$. A Figura 24 apresenta uma biclique do grafo da Figura 22 que se encaixa nesse caso. O vértice $v_{10}$ do conjunto $B_4$ é o centro da estrela e as pontas são todos os vértices do conjunto independente.*

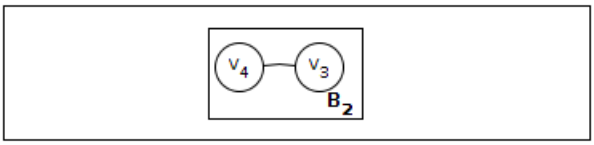
**Figura 24 – Primeiro caso de biclique no *threshold***




**Fonte: Autoria própria**

2. *$D$ é $\{\{v\}\{w\}\}$, com $v$ e $w$ gêmeos fortes (pertencentes ao mesmo bloco $B_i$). A Figura 25 mostra uma biclique do grafo da Figura 22 que obedece essa situação. Essa biclique é $\{\{v_3\}\{v_4\}\}$.*

**Figura 25 – Segundo caso de biclique no *threshold***




**Fonte: Autoria própria**

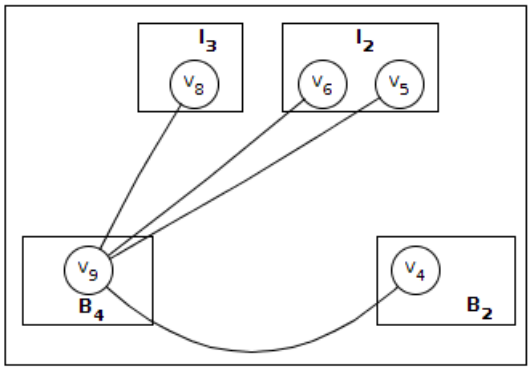
3. *$D$ é uma estrela com centro em $b_j \in B_j$ para $j \in \{2, \ldots, r\}$, onde uma ponta é $b_i \in B_i$ com $i \in \{1, \ldots, j-1\}$, $i \neq j$, e os outros vértices são o conjunto formado por todos os vértices de $I_k$ tal que $i \leq k < j$. A Figura 26 apresenta uma biclique do grafo da Figura 22 que se encaixa nesse caso. A biclique tem centro em $v_9$ e pontas no conjunto independente e em $v_4$ que é vértice da clique.*

Note que cada $B_i$ é um bloco, então, para algum $i$, $\beta(G) = B_i$. Assim é possível determinar se existem uma $k$-biclique coloração de um *threshold* $G$ com $k$ cores onde $k = \beta(G)$ com base nas condições do Teorema 4.8.

**Teorema 4.8.** *(TERLISKY, 2010) Seja $G$ um grafo threshold tal que $\beta(G) = k$, e $B_1, I_1, B_2, I_2, \ldots, B_r, I_r$ sua ordenação threshold. Então é equivalente:*

- *O número biclique cromático de $G$ é $k$.*
- *Não existem $i$ e $j \in \{1, \ldots, r\}$ tais que $i < j$, $|B_i| = |B_j| = k$, $|B_q| = k - 1$ e $|I_h| = 1$ para todo $q \in \{i+1, \ldots, j-1\}$ e todo $h \in \{i, \ldots, j-1\}$.*

**Figura 26 – Terceiro caso de biclique no *threshold***



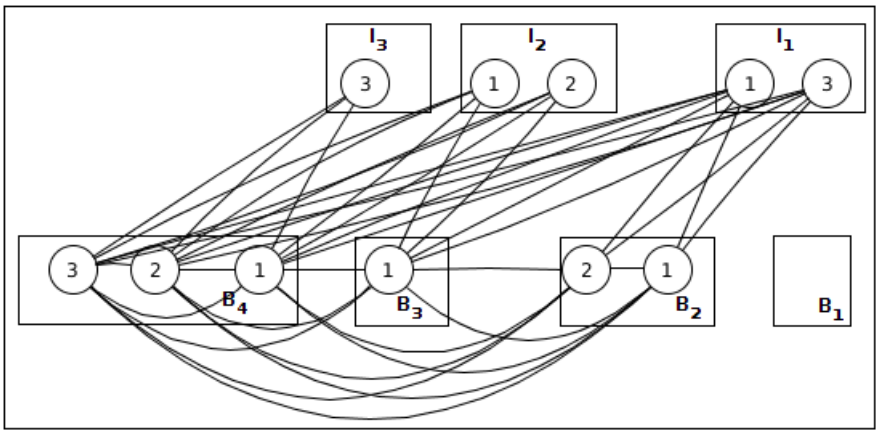

**Fonte: Autoria própria**

Os passos para realizar a coloração de um grafo *threshold* tais que quaisquer $B_i$ e $B_j$ satisfazem as condições do Teorema 4.8 são os seguintes:

- Colorindo $B_i$: para cada $i \in \{i, \ldots, r\}$ colorir os vértices do bloco $B_i$ utilizando as cores $1, 2, \ldots, |B_i|$.
- Colorindo $I_i$: para cada $i \in \{i, \ldots, r-1\}$, se o bloco $B_{i+1} = k$, colorir um vértice de $I_i$ com cor $k$. Se $B_{i+1} = q < k$, colorir um vértice de $I_i$ com a cor $q+1$. Atribuir aos demais vértices de $I_i$ a cor 1.

A Figura 27 mostra um exemplo de aplicação da coloração para um grafo que obedece as propriedades do Teorema 4.8.

**Figura 27 – Exemplo coloração em *threshold***




**Fonte: Autoria própria**

A partir do Teorema 4.8 é possível desenvolver um algoritmo que devolve a coloração biclique ou $no$, dado um grafo *threshold* $G$ e um número $k$ de cores. O Teorema 4.9 enuncia a complexidade desse algoritmo.

**Teorema 4.9.** *(TERLISKY, 2010) O Problema da Coloração Biclique para grafos threshold pode ser resolvido em tempo $O(n)$, se são dados os graus dos vértices como entrada e em tempo $O(n + m)$, caso contrário.*

No algoritmo, após montar a sequência de conjuntos $B_1, I_1, B_2, I_2, \ldots, B_r$, é feito o seguinte: se o bloco $B_x$ tem tamanho $k$, é mantido um registro para verificar se $B_x$ é o começo de uma subsequência de tamanho $(k, 1, k-1, 1, \ldots, k-1, 1, k)$. Se isso acontecer, ou aparecer bloco $B_y$ de tamanho maior que $k$, não é possível colorir o grafo em coloração biclique com $k$ cores, então é devolvido não como resposta. Caso contrário é aplicada a coloração como descrita no Teorema 4.8 e retornado o grafo em coloração biclique.

# 5 NOVOS RESULTADOS EM COLORAÇÃO BICLIQUE

Esse capítulo descreve um algoritmo eficiente para a determinação do número biclique cromático em uma subclasse dos cografos definida no contexto desse trabalho de conclusão de curso e chamada de Cografos de União com Pendente.

## 5.1 COGRAFO DE UNIÃO COM PENDENTE

A Definição 5.1 descreve a operação de união com pendente.

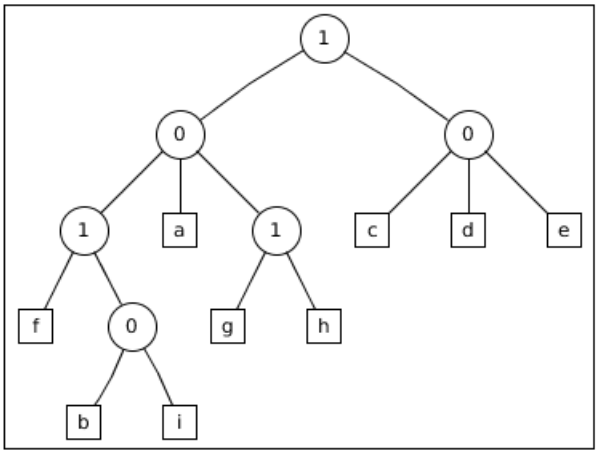
**Definição 5.1** (União com pendente)**.** *Sejam $G_1, G_2, G_3, \ldots, G_k$ grafos quaisquer. A operação de união com pendente $\cup^{\square}(G_1, G_2, G_3, \ldots, G_k)$ é definida por $G_1 \cup G_2 \cup G_3 \cup \cdots \cup G_k \cup K_1$.*

A Definição 5.2 estabelece a subclasse dos cografos de união com pendente.

**Definição 5.2** (Cografo de União com Pendente)**.** *Um cografo $G$ é cografo de união com pendente se:*

*(i) é um grafo $K_1$;*
*(ii) é um grafo $\cup^{\square}(G_1, G_2, G_3, \ldots, G_k)$, tal que $G_i$ é um cografo de união com pendente, $1 \leq i \leq k$;*
*(iii) é um grafo $G_1 * G_2$, onde $G_1$ e $G_2$ são cografos de união com pendente.*

**Figura 28 – Cografo de união com pendente**




**Fonte: Autoria própria**

Seja $G$ um cografo de união com pendente e $T$ sua coárvore canônica. A partir da Definição 5.2, observa-se que $T$ possui em cada nó $0$ pelo menos um filho que é vértice. A Figura 28 mostra uma coárvore de um cografo de união com pendente. Note que todos os nós rotulados com $0$ possuem ao menos um filho que é vértice, o que caracteriza essa subclasse.

Pela garantia de pelo menos um vértice pendente em cada nó rotulado com $0$ em cografos de união com pendente, a policromaticidade das bicliques é mais simples de ser garantida se comparado ao caso geral. A coloração biclique em cografos de união com pendente é demonstrada na próxima seção.

## 5.2 COLORAÇÃO BICLIQUE EM COGRAFOS DE UNIÃO COM PENDENTE

Nessa seção apresentamos como o Problema da Coloração Biclique foi resolvido na subclasse dos cografos de união com pendente.

A Propriedade 4.3 define que o limite inferior para a coloração biclique de um grafo $G$ é $\chi_{bc}(G) \geq \beta(G)$. Dado um cografo $G$, para determinar $\beta(G)$ fazemos uso da coárvore como mostra o Lema 5.3. Relembre que um bloco é um conjunto maximal de gêmeos fortes, ou seja, um conjunto de vértices que têm a mesma vizinhança fechada. O número de bloco $\beta$ é o tamanho do bloco máximo do grafo.

**Lema 5.3** (Número de bloco em cografos)**.** *Dado um cografo $G$ e sua coárvore canônica $T$, $\beta(G)$ é o máximo entre $1$ e o maior número de vértices filhos de um nó rotulado com $1$ em $T$.*

*Demonstração.* Pelo Lema 3.4, o subconjunto de descendentes de exatamente duas subárvores enraizadas em dois filhos distintos $x$ e $y$ de um nó $v$ rotulado com $1$ formam uma biclique. Se $x$ e $y$ são folhas da coárvore e, portanto, vértices do cografo $G$, então a biclique por eles formada só contém uma aresta. Pelo Lema 4.2 uma biclique contém uma aresta se e somente se seus vértices são gêmeos fortes. Logo, $x$ e $y$ são gêmeos fortes. Mais ainda, o conjunto de todos os filhos de $v$ que são folhas são gêmeos fortes. Pela definição de bloco, o conjunto dos filhos de $v$ que são folhas em $T$ constituem um bloco em $G$. Observe que esse bloco é maximal. De fato, suponha por contradição que existe um vértice $w_1$ que pertence ao mesmo bloco que $x$ e $y$, tal que $w_1$ é descendente de um nó $u$ filho de $v$. Então $u$ é um nó $0$ e existe pelo menos um vértice $w_2$ descendente de $u$ e não adjacente a $w_1$. Mas $w_2$ é adjacente a $x$ e $y$, por construção, contradizendo a hipótese que $x$, $y$ e $w_1$ pertencem ao mesmo bloco. Portanto, o tamanho do bloco máximo é o maior número de filhos vértices em um nó $1$. Se não existem $x$ nem $y$ que sejam vértices filhos de um mesmo nó $1$, então $G$ não tem vértices gêmeos fortes e cada vértice forma um bloco de tamanho máximo. Logo, o maior bloco tem tamanho $1$. □

**Lema 5.4** (Bicliques em cografos de união com pendente)**.** *Se $G$ é um cografo de união com pendente, então bicliques que não contém vértices filhos de nós rotulados com $0$ são bicliques de uma única aresta.*

*Demonstração.* Suponha que existe uma biclique $Q$ tal que todos os seus vértices são filhos de nós rotulados com 1. Então, os vértices de um mesmo conjunto independente em $Q$ são filhos de nós com rótulo 1 dois a dois distintos. Sejam $w_1$ e $w_2$ dois vértices do mesmo conjunto

independente em $Q$. Seja $p_1$ o pai de $w_1$ e $p_2$ o pai de $w_2$. Como $w_1$ e $w_2$ pertencem ao mesmo conjunto independente em $Q$, $p_1$ e $p_2$ tem um ancestral comum mais próximo rotulado com $0$ na coárvore, chamado $V$. Como $G$ é um cografo de união com pendente, $V$ tem um filho $y$ que é folha na coárvore, ou seja, é um vértice de $G$. Pelo Lema 3.2, qualquer conjunto independente maximal enraizado em $V$ contém obrigatoriamente $y$. Então, $w_1$, $w_2$ e $y$ pertencem ao mesmo conjunto independente maximal da biclique que, portanto, tem mais que uma aresta.

□

Como consequência do Lema 5.4, em cografos de união com pendente, cada biclique ou é formada por uma única aresta ou contém pelo menos um vértice que é filho de nó rotulado com $0$. Para bicliques de uma única aresta sabemos, pelo Lema 5.3, que são formadas por vértices filhos de um mesmo nó $1$. Por isso, vértices irmãos filhos de nó $1$ não podem ter a mesma cor. Assim, para obter uma coloração biclique em vértices filhos de nós rotulados com $1$, deve-se iniciar atribuindo ao primeiro vértice a cor $1$ e incrementar a cor a cada vértice irmão. Com isso é garantido que bicliques com apenas uma aresta não são monocromáticas.

Antes de determinar a forma de coloração para bicliques com mais de dois vértices, vamos descrever as maneiras de obter bicliques policromática olhando conjuntos independentes maximais que as formam.

Uma biclique $Q$ com conjuntos independentes maximais $A$ e $B$ não é monocromática quando $A$ ou $B$ são policromáticos, ou $A$ e $B$ são monocromáticos, mas a cor usada em $A$ é distinta da usada em $B$. Por exemplo, para os vértices de $A$ é atribuída cor $1$ e cor $2$ para os vértices de $B$.

Seja $X$ o nó em que a biclique $Q$ está enraizada e $n_0$ um nó $0$ filho de $X$ que possui descendentes em $Q$, há duas maneiras de garantir que um conjunto independente maximal enraizado em $n_0$ é policromático sabendo que, pelo Lema 3.2, pelo menos um descendente de cada filho de $n_0$ será incluído no conjunto independente maximal. Na primeira maneira, $n_0$ possui pelo menos dois filhos folhas, $x$ e $y$. Nesse caso, atribuindo cores distintas a $x$ e $y$ todo conjunto independente que contém descendentes de $n_0$ é policromático.

Antes de ver o segundo caso de conjunto independente maximal policromático faz-se necessário entender os casos de conjuntos independentes maximais monocromáticos que formam bicliques policromáticas.

Seja $T(X)$ a subárvore onde a biclique $Q$ está enraizada. Pelo Lema 3.4, $X$ é nó rotulado com $1$ e duas subárvores, $T(W)$ e $T(Z)$, enraizadas em filhos de $X$ formam $Q$, onde cada subárvore possui descendentes em um conjunto independente maximal. Então, há os seguintes casos envolvendo conjuntos independentes maximais monocromáticos que formam bicliques policromáticas:

- Se $W$ e $Z$ são folhas. Neste caso, $W$ e $Z$ formam uma biclique de uma única aresta. Então, cada biclique de uma única aresta, então cada conjunto independente maximal contém

apenas um vértice e, portanto, é monocromático. Para que a biclique seja policromática, para cada folha filha de $X$ é atribuída uma cor distinta iniciando em 1.

- Se $W$ é o nó 0 no qual está enraizado $A$, um conjunto independente maximal monocromático, e $Z$ é uma folha. O nó $W$ possui apenas um filho folha $y$, já que, se possuísse dois ou mais filhos folhas $A$ não seria monocromático atribuindo cores distintas a esse vértices. É preciso que $y$ possua cor distinta de $Z$ para a biclique não ser monocromática. Seja $F_{n_1}$ o conjunto dos $i$ filhos folhas de $X$, onde $i$ é o número de filhos folhas, para os quais foram atribuídas as cores de 1 a $i$. Para que nenhuma biclique formada por $A$ e $Z$ seja monocromática, $y$ não pode ter nenhuma das $i$ cores. Logo, a menor cor disponível para $y$ é $i + 1$.
- Se $W$ e $Z$ são nós 0s, com conjuntos independentes maximais monocromáticos $A$ e $B$, respectivamente, $y$ filho folha de $W$ e $o$ filho folha de $Z$. É preciso que $y$ e $s$ possuam cores distintas para a biclique ser policromática. Considere que $y$ e $s$ possuem cor $j$, pelo item anterior sabemos que $j$ não pode ser uma cor já utilizada em $F_{n_1}$. Considere que $j$ é a cor $i + 1$. Sem perda de generalidade considere que trocaremos a cor de $s$. A menor cor possível para $s$ é $j+1$, pois qualquer cor até $j-1$ resultará em uma biclique monocromática formada por $B$ e um vértice de $F_{n_1}$. Note que $y$ se tornou um falso filho de $X$, pois foi utilizada a cor de $y$ aumentada em um para colorir $s$. Da mesma forma, a cor de $s$ também se tornará um falso filho de $X$, e se outro nó 0 for raiz de um conjunto independente maximal monocromático, a cor de seu filho será a cor de $s$ aumentada em 1.

Note que o número de filhos folhas de um nó $X$ rotulado com 1 somado ao número de falsos filhos folhas, resulta no maior valor possível de bicliques monocromáticas enraizadas em $X$.

Voltando aos casos de conjuntos independentes maximais monocromáticos, esse segundo caso considera que $n_0$ possui apenas um filho folha $y$. Como $y$ é o único filho folha, existe pelo menos um nó $W$ rotulado com 1 que é filho de $n_0$. Considere que todas as bicliques enraizadas em $W$ são policromáticas. Então o número de filhos folhas somado ao número de falsos filhos é o valor $f$ que é o maior valor possível de bicliques monocromáticas enraizadas em $W$. Atribuindo $f + 1$ a $y$, esse vértice terá cor distinta de qualquer conjunto independente maximal monocromático em $T(W)$, resultando que em $T(n_0)$ os conjuntos independentes maximais são policromáticos.

Sempre é possível aplicar essa forma de coloração para $y$ partindo das cores dos filhos e falsos filhos dos nós 1 irmãos. Entretanto, há casos que se a coloração for feita dessa forma aumentará o número de cores necessárias. Para tentar usar apenas $\beta$ cores na coloração, a cor de $y$ pode se basear nas cores dos filhos e falsos filhos da raiz da biclique, pois essas dizem qual é a maior cor que torna bicliques monocromáticas. Entretanto, há casos em que serão utilizadas $\beta + 1$ cores, ou seja, um cografo de união com pendente $G$ tem número biclique cromático $\chi_{bc}(G)$ tal que $\beta(G) \leq \chi_{bc}(G) \leq \beta(G) + 1$.

Dada uma coárvore $T$ de um cografo de um união com pendente $G$ a coloração biclique mínima é feita como apresentado nos itens abaixo.

- Filhos de nós 1: atribua cores aos vértices iniciando em 1. No pior caso serão usadas $\beta$ cores como enunciado no Lema 5.3 e todas as bicliques de uma aresta não serão monocromáticas.
- Filhos de nós 0 com mais de um irmão vértice: atribua cor 1 a um deles e 2 aos demais e toda biclique que contém descendentes desse nó 0 não será monocromática. A coloração será mínima, pois são necessárias pelo menos duas cores para uma biclique não ser monocromática.
- Filho vértice $y$ único de $X$ que é nó 0: primeiro são coloridas todas as subárvores enraizadas em nós 1 filhas de $X$. Se para algum nó $w$ rotulado com 1, irmão de $y$, a última cor utilizada para seus filhos e falsos filhos é menor que $\beta$, atribua a próxima cor a $y$. Nesse caso, $y$ terá no máximo cor $\beta$, não aumentado o número de cores necessárias para a coloração biclique do cografo. Seja $R$ o pai de $X$. Se todos os filhos nós 1 tem número de filhos igual a $\beta$, então verifique a soma do número de filhos folhas e falsos filhos folhas de $R$. Se esse valor for menor que $\beta$ atribua a próxima cor para $y$ e aumente em um o número de falsos filhos de $R$. Assim, o conjunto independente maximal enraizado em $T(X)$ será monocromático, entretanto, qualquer outro conjunto independente maximal enraizado nos filhos de $R$ forma uma biclique não monocromática com os conjuntos independentes enraizados em $X$, devido a cor de $y$. Caso a última cor usada nos filhos folhas e falsos filhos folhas de $R$ seja igual a $\beta$ a cor de $y$ deve ser $\beta + 1$. Isso aumentará o número de cores necessárias para a coloração em 1, mas não há como usar menos cores, pois há conjuntos independentes maximais monocromáticos enraizados em $T(X)$ para qualquer valor de 1 a $\beta$ e há conjuntos independentes em outras subárvores de $R$ monocromáticas para valores de 1 a $\beta$.

Com esses passos é possível criar um algoritmo para coloração biclique em cografos de união com pendente. A estrutura dos nós de operação, para esse algoritmo é baseada na estrutura usada nos nós da coárvore gerada no algoritmo de reconhecimento de cografos com o acréscimo de dois campos, campo $pai$, uma referência para o nó pai, e numeroFilhosVertices, que guarda a maior cor com a qual há conjuntos independentes maximais monocromáticos enraizados naquele nó, se o no for rotulado com 1, ou seja última cor usada em filhos e falsos filhos de nós 1.

Antes do algoritmo de coloração é necessário executar uma busca em profundidade no grafo para encontrar o valor de $\beta$, ou seja, para contar qual é o maior número de filhos folhas em um mesmo nó 1. O valor de $\beta$ é uma constante global para o algoritmo de coloração. Nessa busca também é preenchido o valor do campo numeroFilhosVertices para todos os nós.

O Algoritmo 8 colore qualquer cografo de união com pendente com o mínimo de cores necessárias, ou seja, $\beta$ ou $\beta + 1$ cores.

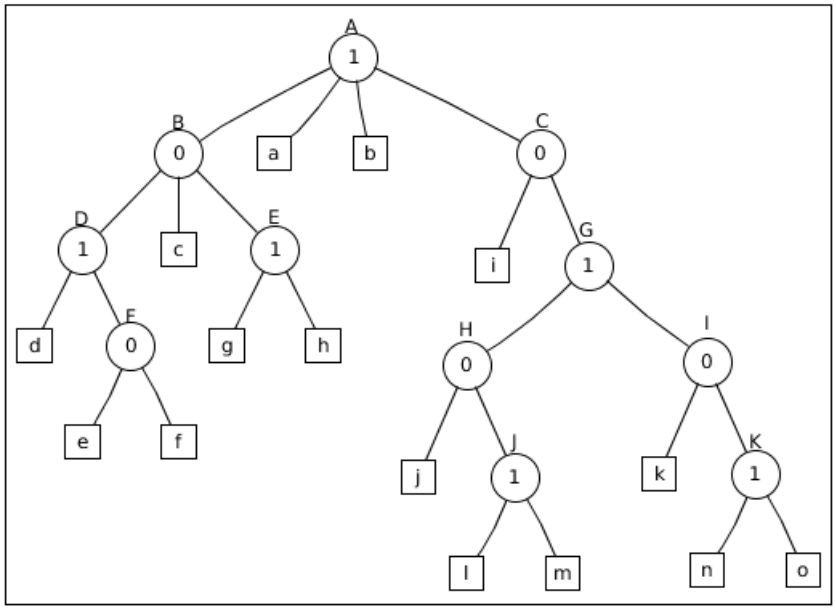
**Algoritmo 8:** colorirCografoUniaoPendente(nó No)

```
Entrada: Estrutura do nó de operação 0 ou 1
Saída: última cor usada se nó é do tipo 1 ou 0 caso contrário
1  início
2      se No→tipo == 1 então
3          cor ⟵ 1
4          para Cada vértice v da listaFilhosVertices faça
5              v ⟵ cor
6              cor ⟵ cor +1
7          fim
8          para Cada nó M da listaFilhosNos faça
9              colorirCografoUniaoPendente(M)
10         fim
11         retorna numeroFilhosVertices
12     senão
13         se Possui apenas 1 filho vértice v então
14             menor ⟵ β + 1
15             para Cada nó M da listaFilhosNos faça
16                 aux ⟵ colorirCografoUniaoPendente(M)
17                 se menor>aux então
18                     menor ⟵ aux;
19                 fim
20             fim
21             se menor +1 ≤ β então
22                 v ⟵menor +1
23             senão
24                 se No→pai→numeroFilhosVertices < β então
25                     v⟵No→pai→numeroFilhosVertices+1
26                     no→pai→numeroFilhosVertices ⟵ no→pai→numeroFilhosVertices +1
27                 senão
28                     v⟵ β + 1
29                 fim
30             fim
31         senão
32             No→listaFilhosVertices[0] →cor = 1
33             para Cada vértice v da listaFilhosVertices iniciando na posicao 1 faça
34                 v ⟵ 2
35             fim
36             para Cada nó M da listaFilhosNos faça
37                 colorirCografoUniaoPendente(M)
38             fim
39         fim
40     fim
41     retorna 0
42 fim
```

**Fonte: Autoria própria**

Para exemplo de execução, vamos aplicar o Algoritmo 8 na coárore da Figura 29, sabendo que $\beta$ do grafo é 2.

**Figura 29 – Coloração em cografo de união com pendente**



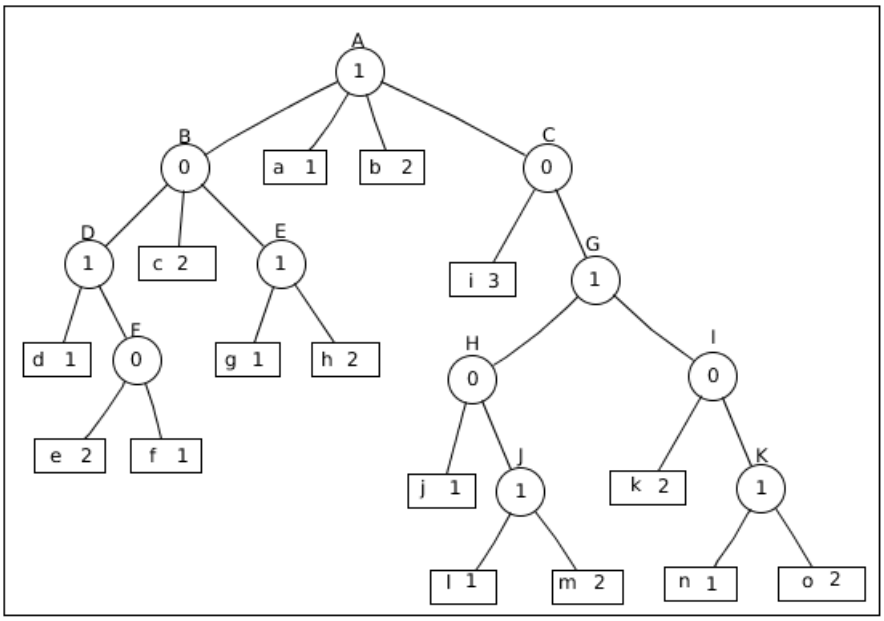

**Fonte: Autoria própria**

A raiz $A$ é recebida na primeira chamada, e são atribuídas as cores 1 e 2, respectivamente, aos vértices $a$ e $b$. Na linha 8, o algoritmo é chamado enviando primeiro $B$ e depois $C$ como parâmetro. Quando $B$ é enviado como parâmetro a linha 13 retorna verdadeiro, então o algoritmo é chamado para os filhos $D$ e $E$ de $B$ comparando o retorno para guardar o menor possível. Na linha de execução de $D$ é atribuída cor 1 para $d$ e chamado o algoritmo enviando $F$ como parâmetro. Para $F$, a primeira condição na linha 13 dá falso e é executado o senão da linha 31. Nele é atribuído cor 1 para $f$ e cor 2 para $e$ e o algoritmo retorna para a chamada do nó $D$ na linha 10. Na linha 11, o algoritmo retorna 1. Voltando a linha 16 da chamada de $B$ é atribuído 1 ao auxiliar $aux$. Como 1 é menor que $\beta + 1$, o valor da variável $menor$ passa a ser 1. O processo é semelhante para a chamada do algoritmo enviando $E$, mas nesse caso o algoritmo retorna 2 e, portanto, o valor de $menor$ não é atualizado. Como $menor$ tem valor menor que $\beta$, o vértice $c$ recebe cor $menor + 1$ que é 2.

Nesse ponto o algoritmo está na linha de execução do nó $A$ na linha 9, onde $C$ é enviado como parâmetro em outra chamada. Por sua vez, $C$ chama o algoritmo para $G$ que, como não tem filhos vértices folha, chama o algoritmo para $H$, que chama o algoritmo para $J$. Nesse ponto, o laço da linha 4 atribui cor 1 para $l$ e 2 para $m$, retornando valor 2. Como $2 + 1$ é maior que $\beta$, a linha 24 verifica se a variável numeroFilhosVertices do pai de $H$, nó $G$, é menor que $\beta$. Como o valor verificado em numeroFilhosVertices é 0, $j$ recebe cor 1 e aumenta em um o valor de numeroFilhosVertices de $G$ que passa a ser 1. Um processo semelhante acontece com o nó $I$, que atribui cor 2 para $k$ também aumentando o valor de numeroFilhosVertices, que agora é 2. Voltando para a linha de execução do nó $C$, este verifica que o valor retornado do nó $G$ é 2 que é igual a $\beta$, então é feita a verificação no vértice $A$, que também possui numeroFilhosVertices igual a $\beta$. Devido a isso, $i$ recebe valor $\beta + 1$.

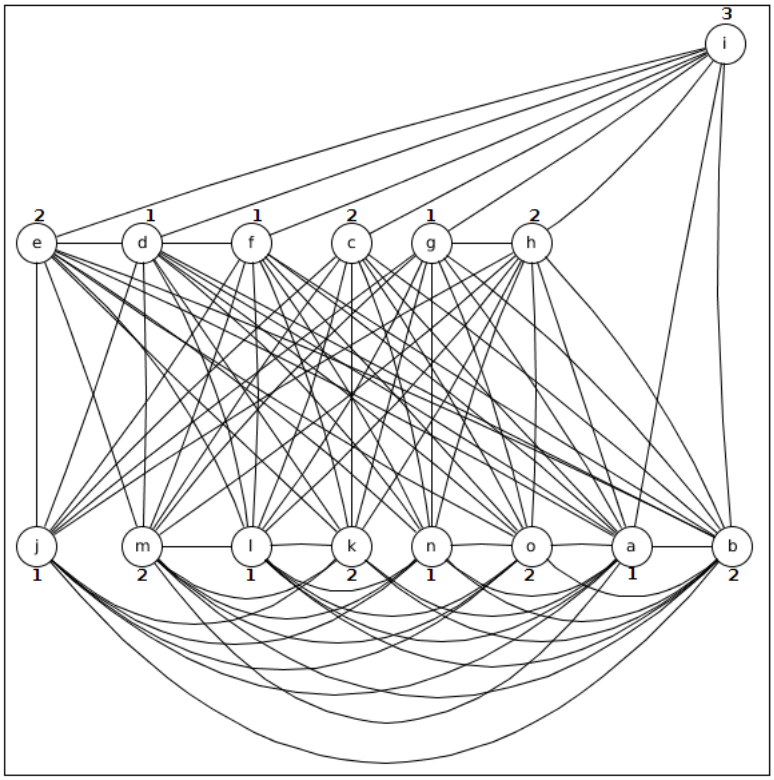
A Figura 30, mostra a coárvore da Figura 29 colorida e a Figura 31 apresenta o grafo representado pela coárvore.

**Figura 30 – Coloração em cografo de união com pendente final**




**Fonte: Autoria própria**

**Figura 31 – Cografo de união com pendente colorido**




**Fonte: Autoria própria**

Com o Algoritmo 8 qualquer cografo de união com pendente pode ser colorido em coloração biclique utilizando o mínimo de cores. Na próxima seção é dada a complexidade

desse algoritmo.

## 5.3 ANÁLISE DA COMPLEXIDADE DO ALGORITMO

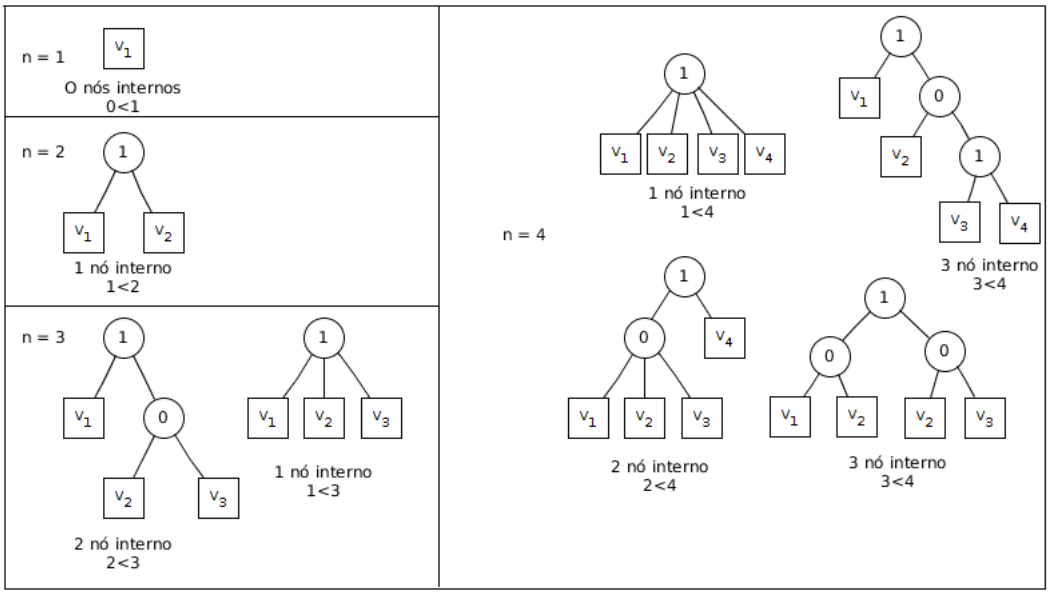
Nessa seção faremos uma discussão sobre a complexidade do Algoritmo 8 em função do número de operações da coárvore, ou seja, os nós internos, considerando o pior caso.

**Lema 5.5.** *Se $T$ é uma árvore com $n$ folhas tal que todo nó interno tem pelo menos dois filhos, então $T$ tem no máximo $n-1$ nós internos.*

*Demonstração.* Vamos provar por indução.

- *Caso base:* Se $n = 1$, ou seja, se a árvore tem apenas uma folha, então não pode ter raíz, pois a raíz teria apenas um filho, contradizendo a hipótese. Logo, $T$ não tem nós internos. Se $n = 2$, então a única possibilidade é que a árvore seja um $P_3$. Logo, tem apenas um nó interno. Para os casos $n = 3$ e $n = 4$, todas as possíveis árvores que satisfazem a hipótese podem ser vistas na Figura 32 e, por inspeção, verifica-se que o número de nós internos é menor que o número de folhas.

**Figura 32 – Exemplo de geração de coárvore**




**Fonte: Autoria própria**

- *Hipótese de indução:* Qualquer árvore com até $k$ folhas tal que todo nó interno tem pelo menos dois filhos é uma árvore com no máximo $k-1$ nós internos.

- *Passo:* Seja $T$ uma árvore com $k+1$ folhas, $k \geq 4$. Se existe um nó $u$ que é pai de três ou mais folhas, então remova uma folha de $u$. A nova árvore $T'$ tem o mesmo número de nós internos que $T$. Além disso $T'$ tem no $k$ folhas e, por hipótese de indução, tem no máximo $k-1$ nós internos. Logo, $T$ tem no máximo $k-1$ nós internos, está provado.

Agora, suponha que todo nó interno é pai de no máximo duas folhas (além de outros possíveis filhos que são nós internos). Como $k \geq 4$, existem pelo menos $5$ folhas. Como não há nó que seja pai de mais de duas folhas e todo nó interno tem pelo menos dois filhos, então existe um nó interno $u$ cujos filhos são exatamente duas folhas. Observe que $u$ não é pai de nós internos. Remova de $T$ os dois filhos de $u$, criando a árvore $T'$. Note que, em $T'$, o nó $u$ é uma folha. Então, $T'$ não tem as duas folhas filhas de $u$, mas tem uma nova folha (o próprio $u$). Então, $T'$ tem $k$ folhas. Pela hipótese de indução, $T'$ tem no máximo $k-1$ nós internos. Como $T$ tem um nó interno a mais (o vértice $u$), então $T$ tem no máximo $k$ nós internos.

□

Antes da execução do Algoritmo 8, deve-se criar uma coárvore canônica para o cografo $G$. Como visto na Seção 2.1, a criação de uma coárvore canônica tem complexidade de tempo $O(|V(G)| + |E(G)|)$. Uma vez que se tenha a coárvore canônica, seu número de nós é no máximo $2n-1$, onde $n$ é o número de vértices do cografo. Como em qualquer árvore $T$ com $|V(T)|$ vértices o número de arestas é $|V(T)| - 1$, temos que o número de arestas dessa coárvore é, no máximo, $2n - 2$. Para determinar o número de bloco desse cografo, é executada uma busca em profundidade na coárvore. A complexidade do algoritmo de busca em profundidade é conhecidamente $O(|V(G)| + |E(G)|)$, para qualquer grafo $G$. O Algoritmo 8 é chamado para cada nó. O número de operações em cada chamada é delimitado por uma constante. Seja $p$ o maior número máximo de operações gastos em uma chamada, no pior caso, serão gastos $p(2n - 1)$ operações, ou seja, o algoritmo é linear. Como o algoritmo de criação da coárvore, a busca em profundidade e o algoritmo de coloração são executados em tempo linear, a complexidade de todo o processo é linear.

# 6 CONCLUSÃO

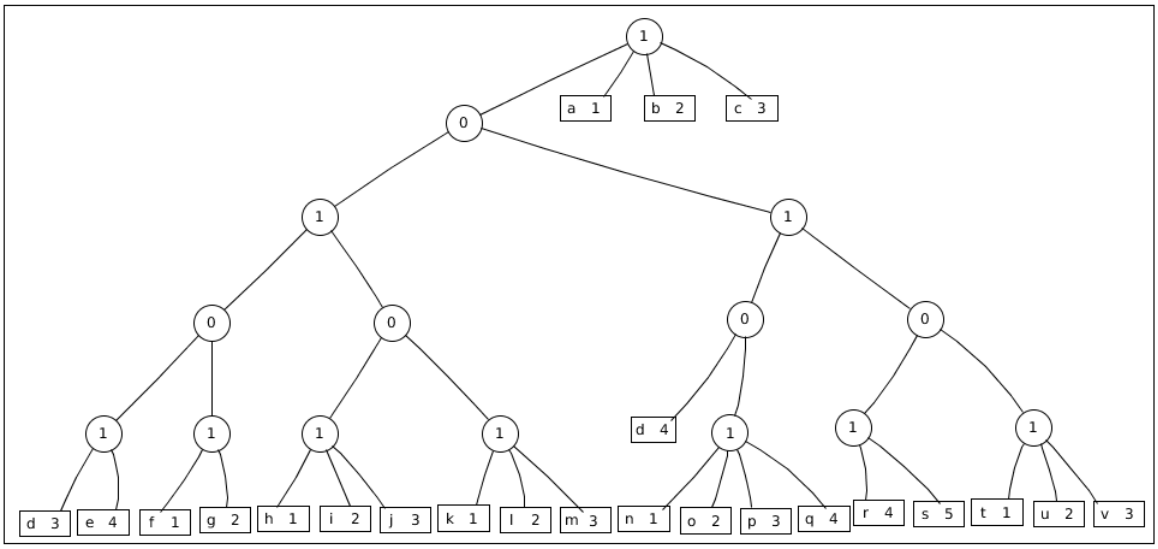
Este trabalho apresentou algoritmos para enumeração e contagem de bicliques dada uma coárvore. Por meio da coárvore também foi possível encontrar o limite inferior $\beta$ da coloração biclique.

Nos cografos de união com pendente, foi verificado que utilizando a coárvore é possível apresentar uma coloração biclique mínima sem a necessidade de listá-las. Nesse trabalho, foi apresentado um algoritmo linear de coloração biclique para essa classe, determinando-se que esses grafos têm $\chi_{bc} \in \{\beta, \beta + 1\}$.

Foi observado que toda biclique em um cografo de união com pendente ou é biclique de uma única aresta ou possui pelo menos um vértice que é filho de nó rotulado com $0$ no grafo original. Além disso, na coloração biclique de cografos de união com pendente é possível garantir a não monocromaticidade das bicliques avaliando-se no máximo três níveis da coárvore para atribuir cor a cada vértice.

A Figura 33 mostra um cografo que não é de união com pendente. Note que para garantir a não monocromaticidade das bicliques toda a árvore precisa ser analisada. Por exemplo, a não monocromaticidade da biclique $\{\{c\}, \{j, m, v, r\}\}$ é garantida no quinto nível iniciando a partir do nó $1$ onde a biclique foi enraizada (nível $0$).

**Figura 33 – Exemplo de coloração no caso geral**




**Fonte: Autoria própria**

Como trabalho futuro sugere-se investigar a existência de algoritmos polinomiais para a coloração biclique de cografos que não são união com pendente. Outro trabalho futuro consiste em provar que o número de bicliques em cografos é exponencial a partir da análise do Algoritmo 7.

# REFERÊNCIAS


ATLURI, Gowtham *et al.* Discovering coherent value bicliques in genetic interaction data. In: **In Proceedings of 9th International Workshop on Data Mining in Bioinformatics (BIOKDD'10**. [S.l.: s.n.], 2000.

BONDY, J.A.; MURTY, U.S.R. **Graph theory (graduate texts in mathematics)**. [S.l.]: Springer New York, 2008.

BRANDSTÄDT, Andreas; LE, Van Bang; SPINRAD, Jeremy P. **Graph classes: a survey**. [S.l.]: SIAM, 1999.

BRETSCHER, Anna *et al.* A simple linear time lexbfs cograph recognition algorithm. **SIAM Journal on Discrete Mathematics**, SIAM, v. 22, n. 4, p. 1277–1296, 2008.

BU, Dongbo *et al.* Topological structure analysis of the protein–protein interaction network in budding yeast. **Nucleic acids research**, Oxford University Press, v. 31, n. 9, p. 2443–2450, 2003.

CHARTRAND, G.; ZHANG, P. **Chromatic graph theory**. [S.l.]: CRC press, 2008.

CORNEIL, D. G.; LERCHS, H.; BURLINGHAM, L. S. Complement reducible graphs. **Discrete Applied Mathematics**, Elsevier, v. 3, n. 3, p. 163–174, 1981.

CORNEIL, Derek G.; PERL, Yehoshua; STEWART, Lorna K. A linear recognition algorithm for cographs. **SIAM Journal on Computing**, SIAM, v. 14, n. 4, p. 926–934, 1985.

DAILEY, David P. Uniqueness of colorability and colorability of planar 4-regular graphs are np-complete. **Discrete Mathematics**, Elsevier, v. 30, n. 3, p. 289–293, 1980.

EPPSTEIN, David. Arboricity and bipartite subgraph listing algorithms. **Information processing letters**, Elsevier, v. 51, n. 4, p. 207–211, 1994.

FILHO, H. B. Macêdo *et al.* Biclique-colouring verification complexity and biclique-colouring power graphs. **Discrete Applied Mathematics**, Elsevier, v. 192, p. 65–76, 2015.

FILHO, Hélio B Macêdo; MACHADO, Raphael CS; FIGUEIREDO, Celina MH de. Efficient algorithms for clique-colouring and biclique-colouring unichord-free graphs. **Algorithmica**, Springer, v. 77, n. 3, p. 786–814, 2017.

FROIDURE, Veronique. **Rangs des relations binaires et semigroupes de relations non ambigus**. Tese (Doutorado) — Paris 6, 1995.

GOLUMBIC, Martin Charles. **Algorithmic graph theory and perfect graphs**. [S.l.]: Elsevier, 2004. v. 57.

GROSHAUS, Marina; SOULIGNAC, Francisco J; TERLISKY, Pablo. The star and biclique coloring and choosability problems. **arXiv preprint arXiv:1210.7269**, 2012.

KUMAR, Ravi *et al.* Trawling the web for emerging cyber-communities. **Computer networks**, Elsevier, v. 31, n. 11-16, p. 1481–1493, 1999.

LAWLER, Eugene L.; LENSTRA, Jan Karel; KAN, AHG Rinnooy. Generating all maximal independent sets: Np-hardness and polynomial-time algorithms. **SIAM Journal on Computing**, SIAM, v. 9, n. 3, p. 558–565, 1980.

LERCHS, H. On cliques and kernels. **Department of Computer Science, University of Toronto**, 1971.

LI, Jinyan *et al.* Maximal biclique subgraphs and closed pattern pairs of the adjacency matrix: A one-to-one correspondence and mining algorithms. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 19, n. 12, p. 1625–1637, 2007.

LIMA, A. R. Cunha. **Classificação de Cografos quanto ao índice cromático**. Tese (Doutorado) — Universidade Federal da Fronteira Sul, 2014.

LUBIW, Anna. The boolean basis problem and how to cover some polygons by rectangles. **SIAM Journal on Discrete Mathematics**, SIAM, v. 3, n. 1, p. 98–115, 1990.

NAGARAJAN, Niranjan; KINGSFORD, Carl. Uncovering genomic reassortments among influenza strains by enumerating maximal bicliques. In: IEEE. **Bioinformatics and Biomedicine, 2008. BIBM'08. IEEE International Conference on**. [S.l.], 2008. p. 223–230.

PRISNER, Erich. Bicliques in graphs i: Bounds on their number. **Combinatorica**, Springer, v. 20, n. 1, p. 109–117, 2000.

TERLISKY, Pablo. Biclique-coloreo de grafos. **Master's thesis, Universidad de Buenos Aires (July 2010)**, 2010.

WILLE, Rudolf. Restructuring lattice theory: an approach based on hierarchies of concepts. In: **Ordered sets**. [S.l.]: Springer, 1982. p. 445–470.