
**UNIVERSIDADE FEDERAL DO AMAZONAS**
**INSTITUTO DE COMPUTAÇÃO**
**PROJETO DE INICIAÇÃO CIENTÍFICA**

# SOBRE O PROBLEMA DA LISTA COLORAÇÃO EM GRAFOS

NADNY MACIEL DANTAS DANTAS

Julho de 2015

Manaus - AM

NADNY MACIEL DANTAS DANTAS

# SOBRE O PROBLEMA DA LISTA COLORAÇÃO EM GRAFOS

Relatório Técnico Final de Projeto de Iniciação Científica, organizado pela Fundação de Amparo à Pesquisa do Estado do Amazonas, na Universidade Federal do Amazonas, apresentado como requisito obrigatório definido pelo termo de compromisso.

Orientadora: Rosiane de Freitas Rodrigues, PhD.

Julho de 2015

Manaus - AM

# Sumário

# Lista de Figuras

# Capítulo 1

# Introdução

Este Projeto de Iniciação Científica é continuação do projeto de PIBIC-Jr 2013-2014 (PIBJR-E0001), que visou estudar a Teoria dos Grafos, com foco em Grafos Geométricos, Algoritmos e Aplicações. Neste trabalho atual, o objetivo é investigar uma variação do problema clássico de coloração em grafos, o Problema da Lista Coloração.

O problema de coloração é um dos problemas mais antigos e conhecidos da teoria dos grafos e possui diversas variações. A **coloração de vértices** de um grafo consiste em colorir cada vértice $v$ de um grafo $G = (V, E)$, de tal forma que os vértices adjacentes a $v$ possuam cores diferentes da cor atribuída a $v$. Quando são impostas restrições adicionais sobre as cores disponíveis para cada vértice, surgem variados problemas interessantes para pesquisa científica. Um deles é o **Problema da Lista Coloração**, o tema principal deste trabalho de iniciação científica.

No Problema da Lista Coloração, dado um grafo $G = (V, E)$, existe um conjunto associado $L(v)$ de cores permitidas para cada vértice $v$ de $V(G)$. Este conjunto consiste na lista de cores disponíveis para se colorir o vértice $v$, onde tal vértice pode ser colorido somente pelas cores pertencentes a esta lista. Dessa forma, uma Lista Coloração de $G$ é uma coloração própria de $G$ tal que a cor $c(v)$ atribuída ao vértice $v$ deve pertencer a lista de cores $L(v)$ a ele associada.

Os algoritmos do Problema da Lista Coloração e sua complexidade computacional para classes específicas de grafos, bem como o comportamento das propriedades de lista coloração em algumas subclasses especiais de grafos, além de selecionabilidade *f-choosability* e *k-choosability*, são estudados neste projeto.

Nas seções seguintes serão apresentados o contexto, a motivação geral e objetivos

específicos, a metodologia de pesquisa, o cronograma de atividades e a organização dos capítulos restantes deste documento.

## 1.1 Contexto

Em 2014, após a conclusão do Projeto de Iniciação Científica Júnior 2013-2014, a aluna foi convidada a participar de um Projeto de Iniciação Científica em nível de graduação, seguindo a linha de conhecimento do PIBIC-JR, ou seja, estudando um problema relacionado à Teoria dos Grafos. Dessa forma, iniciaram-se os trabalhos do projeto atual sobre o Problema da Lista Coloração em Grafos, que será concluído no fim de 2015.

## 1.2 Motivação e Objetivos

Nesta seção, são apresentados a motivação geral e os objetivos específicos, que foram estipulados para o desenvolvimento desta iniciação científica.

### 1.2.1 Motivação

A motivação deste projeto é estudar teorias e técnicas para investigar o clássico Problema de Lista Coloração em Grafos, de tal forma a verificar e entender o comportamento de propriedades em subclasses especiais de grafos, organizar os resultados existentes, bem como adaptar e desenvolver algoritmos que o resolvam de forma eficiente.

### 1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho estão descritos a seguir:

- Estudar os conceitos e teoremas de coloração de vértices em grafos, com foco no Problema da Lista Coloração e suas variações;
- Verificar e validar o comportamento das propriedades da lista coloração em subclasses de grafos especiais na tentativa de propor caracterizações para os problemas de investigação;
- Estudar e adaptar modelos de algoritmos existentes para o problema clássico de coloração em vértices, nas variações em estudo;

## 1.3 Metodologia de Pesquisa

Ao longo de um ano de desenvolvimento do trabalho, foram necessárias reuniões periódicas junto à orientadora do projeto para definição e cumprimento de metas. Visto que o projeto de pesquisa atual possui um embasamento teórico semelhante ao projeto anterior, como o estudo da Teoria dos Grafos, muito do que foi estudado em seu desenvolvimento será aproveitado nesta pesquisa.

A metodologia de pesquisa deste projeto consiste nas seguintes etapas:

1. Fazer levantamento de material de pesquisa relativo ao assunto do projeto durante a revisão da literatura.

2. Manter reuniões periódicas com a orientadora para definir metas de desenvolvimento, bem como para obtenção de referencial teórico necessário à compreensão do assunto do projeto.

3. Reunir-se com membros do grupo de pesquisa em Otimização, Algoritmos e Complexidade Computacional com interesses em comum para discutir e aprofundar os conhecimentos sobre o tema da pesquisa.

4. Registrar a pesquisa na forma de relatório técnico parcial, resumos e, por fim, relatório final, incluindo os resultados obtidos durante a pesquisa.

5. Organizar e divulgar os resultados da pesquisa em realização sob a forma de relatórios e artigos científicos.

6. Compartilhar os resultados obtidos com a comunidade de algoritmos e combinatória em eventos nacionais e internacionais.

7. Elaborar o relatório final e preparar a defesa do projeto.

## 1.4 Cronograma de Atividades

A metodologia de pesquisa foi preparada para ocorrer no período de dois anos, de acordo com o cronograma apresentado na Tigura 1.1.

| Atividades | 2014 | | | | | 2015 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Ago | Set | Out | Nov | Dez | Jan | Fev | Mar | Abr | Mai | Jun | Jul |
| 1 | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | |

Concluído Em realização

Figura 1.1: Cronograma de atividades da pesquisa.

## 1.5 Organização do Trabalho

Neste Capítulo 1, foi dada uma introdução composta pelo contexto, definição do problema, objetivos, método de pesquisa proposto e cronograma. No Capítulo 2, será dado sobre Fundamentos Teóricos, serão apresentadas as definições de Teoria dos Grafos e Coloração em Grafos, bem como sobre Teoria da Computação, Otimização Combinatória e Complexidade Computacional. No Capítulo **??**, será dado sobre o Problema de Lista Coloração em Grafos, bem como as principais provas relacionadas e propriedades da selecionabilidade. No Capítulo 4, será dado sobre os Resultados Parciais, com as principais provas estudadas em detalhes e os algoritmos implementados.

E por fim, no Capítulo 5, será dado sobre as Considerações Finais, são feitos os comentários gerais sobre a pesquisa, resumindo as propostas de trabalho para o problema, bem como o andamento e contribuições previstas para a pesquisa.

# Capítulo 2

# Referencial Teórico

Neste capítulo serão apresentados conceitos fundamentais para o desenvolvimento desta pesquisa, envolvendo os principais conceitos em Teoria dos Grafos e Coloração em Grafos, bem como fundamentação de Teoria da Computação e Complexidade computacional.

## 2.1 Teoria dos Grafos

A Teoria dos Grafos é um ramo da Matemática que estuda objetos combinatórios e suas relações. Estes objetos são chamados *grafos* e auxiliam no estudo das relações entre objetos de qualquer tipo. Segundo [22], diferente de muitos dos ramos da Matemática que foram motivados por problemas envolvendo cálculos, movimento, entre outros, o desenvolvimento da Teoria de Grafos se deu através de problemas envolvendo jogos e quebra-cabeças. Atualmente, esta teoria é aplicada em vários ramos da matemática, informática, engenharia e indústria, visto que grafos são um bom modelo para solucionar muitos dos problemas destas áreas.

Esta seção formaliza os conceitos básicos de grafos, bem como algumas de suas classes importantes. As notações utilizadas no restante deste trabalho são introduzidas nesta seção.

### 2.1.1 Conceitos Básicos

Antes de se iniciar um estudo relacionado a grafos, é necessário saber alguns conceitos básicos. Primeiramente, um grafo $G = (V, E)$ é uma estrutura finita não-vazia composta por dois conjuntos: conjunto de vértices $V(G)$ e um conjunto de arestas $E(G)$ [24]. Cada

elemento $e$ no conjunto $E$ é um par $(i, j)$ que indica que o vértice $i$ é adjacente ao vértice $j$, ou seja, $i$ e $j$ são adjacentes e a aresta $e$ incide nos dois. A representação gráfica de um grafo consiste em pontos distintos associados a cada vértice e, para cada aresta $(i, j)$, um segmento de reta conectando os pontos correspondentes aos vértices $i$ e $j$. Quando os pares de vértices que representam as arestas não são ordenados, isto é, $(i, j) = (j, i)$, o grafo é dito não-direcionado. Caso contrário, o grafo é dito direcionado, ou seja, $(i, j) \neq (j, i)$. A Figura 2.1 mostra um exemplo desta representação.

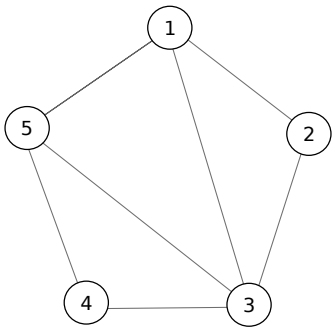



Figura 2.1: Representação gráfica de um grafo não-direcionado com 5 vértices e 7 arestas.

Chama-se **ordem** de $G$ o número de vértices de um grafo $G$ e o número de arestas é o **tamanho** de $G$. O **grau** de um vértice $v$ do grafo é a quantidade de arestas que incidem em $v$. O **grau máximo** do grafo, denotado por $\Delta(G)$, é o valor do maior grau dentre todos os graus dos vértices de $G$. De modo análogo, o valor do menor grau de $G$ é definido como **grau mínimo** e denotado por $\delta(G)$. O conjunto de vizinhos de um vértice, ou seja, o conjunto de vértices adjacentes a $v$ de $G$ é denotado por $N_G(v)$ [8].

**Definição 2.1** (Szwarcfiter, 1986 [24])**.** *Um grafo $H$ é dito ser um **subgrafo** de $G$ se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$.*

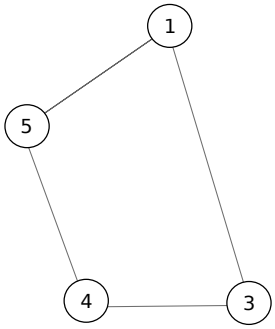



Figura 2.2: Exemplo de subgrafo do grafo $G$ da Figura 2.1.

**Definição 2.2** (Szwarcfiter, 1986 [24])**.** *Seja $H$ um subgrafo de $G$. Se $H$ possuir toda aresta $(uw)$ de $G$ tal que ambos estejam em $V(H)$, então $H$ é o* ***subgrafo induzido*** *pelo subconjunto de vértices de $H$. Diz-se então que $H$ induz $G$.*

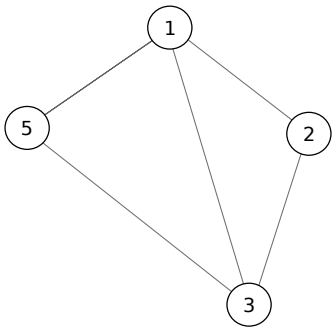



Figura 2.3: Exemplo de subgrafo induzido $H$ do grafo $G$ da Figura 2.1. Logo, $H$ induz $G$.

**Definição 2.3.** *Um* ***grafo completo****, denotado por $K_n$, é aquele em que cada vértice é vizinho de todos os outros vértices, ou seja, todos os vértices são adjacentes entre si.*

Dado um grafo completo $G$ com $n$ vértices, o grau de todos os vértices é exatamente igual a $n-1$. A quantidade total de arestas é dada então por $m = \frac{n(n-1)}{2}$. A Figura 2.4 mostra um exemplo de grafo completo onde $n = 4$, ou seja, um grafo $K_4$.

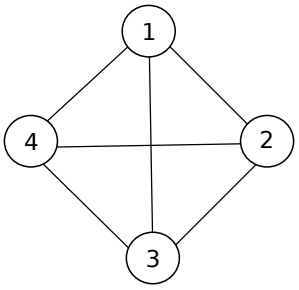



Figura 2.4: Exemplo de grafo completo com $n = 4$ vértices.

**Definição 2.4.** *Uma* ***clique*** *de um grafo $G = (V,E)$ é um subconjunto $V' \subseteq V$ cujo subgrafo induzido $G[V']$ é completo.*

Na Figura 2.5 são indicadas todas as cliques do grafo $G$ da Figura 2.3.

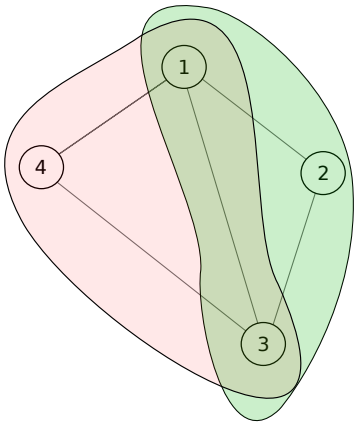



Figura 2.5: Grafo $G$ da Figura 2.3 com as cliques destacadas.

A maior cardinalidade dentre todas as cliques de um grafo $G$ é chamada de **clique máxima**, e o número da clique $\omega(G)$ é o tamanho da clique máxima de $G$.

**Definição 2.5.** *Um **conjunto estável** (ou **independente**) de um grafo $G = (V,E)$ é um subconjunto $V' \subseteq V$ em que nenhum par de vértices é adjacente, para qualquer vértice de $V'$.*

Na Figura 2.6 está indicado um conjunto estável contido no grafo $G$ da Figura 2.3.

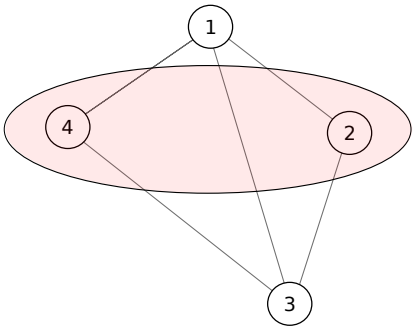



Figura 2.6: Grafo $G$ da Figura 2.3 com as cliques destacadas.

A maior cardinalidade dentre os conjuntos estáveis de um grafo $G$ é chamado de **conjunto estável máximo**, e o número de estabilidade $\alpha(G)$ é o tamanho do conjunto estável máximo de $G$.

### 2.1.2 Classes de Grafos

As definições e características de algumas classes de grafos que serão importantes para este trabalho são destacadas nesta seção.

**Definição 2.6** (Szwarcfiter, 1986 [24])**.** *Um **grafo caminho** (do inglês, path graph) é uma sequência de vértices $P = \{v_1, v_2, ..., v_n\}$ de tal forma que dois vértices são adjacentes se forem consecutivos na sequência, e não adjacentes, caso contrário.*

O comprimento ou tamanho de $P$, denotado por $|P|$, é o número de arestas de $P$, ou seja, $|P| = n - 1$ . Um exemplo de grafo caminho é mostrado na Figura 2.7.

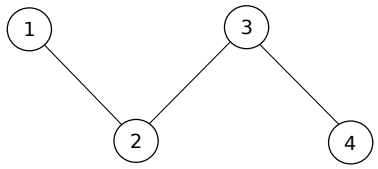



Figura 2.7: Exemplo de um grafo caminho formado por 4 vértices e 3 arestas. Dessa forma, $|P| = 3$.

**Definição 2.7.** *Um **grafo ciclo** (do inglês, cycle graph) $C_n$ (para $n \geq 3$) é um grafo simples cujos vértices podem ser dispostos em uma sequência cíclica de tal forma que dois vértices são adjacentes se são consecutivos na sequência, e não adjacentes em caso contrário. O primeiro e último vértices coincidem.*

Dado um grafo $G$ com $n$ vértices e $k$ arestas, o tamanho de um ciclo é o número de suas arestas. Um ciclo de tamanho $k$ é chamado de $k$-ciclo. Um grafo $G$ é **acíclico** se o mesmo não contém ciclo. Na Figura 2.8 é apresentado um grafo ciclo de $k = 6$.

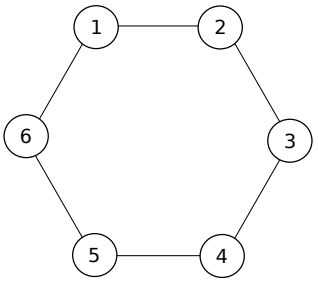



Figura 2.8: Exemplo de um grafo ciclo formado por 6 vértices e 6 arestas. Dessa forma, o tamanho do ciclo é 6.

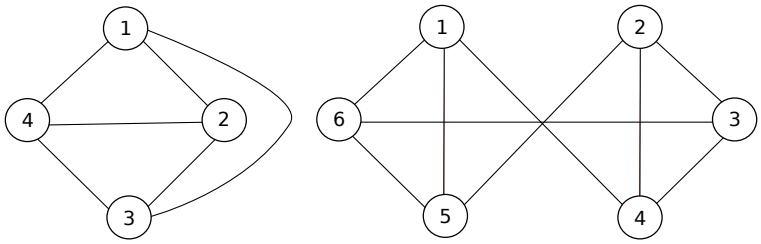
**Definição 2.8.** *Um grafo é **conexo** se quaisquer que sejam os vértices distintos v e u de G, existe sempre um caminho que os conecta. Quando um grafo não é conexo, diz-se que ele é **desconexo**.*

**Definição 2.9.** *Um grafo que possui uma representação gráfica em que nenhuma de suas arestas se cruzem é chamado de grafo **planar**.*

Um grafo planar divide o plano em regiões chamadas de **faces**. A face ilimitada do grafo é chamada de face externa enquanto as outras faces são chamadas de faces internas. Na figura 2.9, um exemplo de grafo planar e um grafo não-planar é dado. O grafo planar de 2.9(a) possui quatro faces, sendo três internas e a face externa.

De acordo com o Teorema 2.1, para determinar a planaridade de um grafo, pode-se usar a famosa fórmula Leonard Euler.

**Teorema 2.1** (da Fórmula de Euler, 1750)**.** *Se G é um grafo conexo e planar então, para uma sua realização plana, verifica-se a igualdade* $|F_0(G)| = |E(G)| - |V(G)| + 2$*, onde* $F_0(G)$ *é o número de faces,* $E(G)$ *é o número de arestas e* $V(G)$ *o número de vértices.*




(a) (b)

Figura 2.9: Exemplo de grafo planar e grafo não-planar. Em (a), um grafo completo de 4 vértices é planar. Em (b), um grafo não-planar, já que $|E(G)| \leq 3|V(G)| - 6$.

**Definição 2.10.** *Um* ***grafo bipartido*** *(do inglês, bipartite graph), é um grafo em que seus vértices podem ser divididos em dois conjuntos disjuntos X e Y tal que uma aresta qualquer tem uma extremidade em X e outra em Y, logo um grafo bipartido é denotado por* $G[X,Y]$.

Se cada vértice em $X$ é ligado a cada vértice em $Y$, então $G[X,Y]$ é um grafo bipartido completo. Se o número de vértices em $X$ é $n$ e o número de vértices em $Y$ é m, o número de arestas de um grafo bipartido completo é $n \times m$. Um exemplo de um grafo bipartido completo $K_{3,3}$ é mostrado na Figura 2.10.

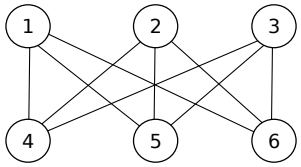



Figura 2.10: Exemplo de um grafo bipartido completo, no qual todos os vértices de $X$ estão ligados a todos os vértices de $Y$. A quantidade de arestas $n \times m$ é igual a 9.

**Definição 2.11** (Andreas, 1999 [19])**.** *Um grafo G é um* ***grafo cordal*** *se cada ciclo em G de tamanho pelo menos* 4 *tem uma corda. Uma corda é uma aresta que conecta dois vértices não adjacentes neste ciclo.*

A Figura 2.9(a) é um exemplo de um grafo cordal. A aresta $\{v_1, v_3\}$ e a aresta $\{v_2, v_4\}$ representam cordas do ciclo $\{v_1, v_2, v_3, v_4, v_1\}$.

**Definição 2.12.** *Um grafo é* ***split*** *se seu conjunto de vértices pode ser particionado em uma clique e em um conjunto estável.*

A Figura 2.11 é um exemplo de grafo split. Os vértices $v_1, v_2$ e $v_3$ fazem parte de um conjunto estável e os vértices $v_4, v_5$ e $v_6$ fazem parte de uma clique.

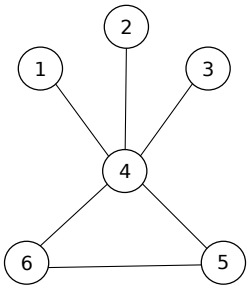



Figura 2.11: Exemplo de um grafo split, formado por um conjunto estável de 3 vértices e uma clique de 3 vértices.

### 2.1.3 O Problema de Coloração em Grafos

Antes de abordar o tema do projeto, o Problema da Lista Coloração em Grafos, é necessário conhecer os conceitos fundamentais de coloração em grafos. Nesta seção, serão apresentados os principais conceitos relacionados a coloração.

**Definição 2.13.** *Uma* ***coloração própria*** *c de vértices em um grafo G é uma atribuição de cores aos vértices de G de tal forma que um vértice v possua cores diferentes dos seus vértices adjacentes.*

As cores são representadas por números inteiros positivos $1, 2, ..., k$. Dessa forma, uma coloração própria pode ser vista como uma função $c : V(G) \to \mathbb{N}$ (sendo $\mathbb{N}$ o conjunto dos números naturais) tal que, se $u$ e $v$ são vértices adjacentes em $G$, temos que $c(v) \neq c(u)$ .

**Definição 2.14.** *Dado um grafo G colorido com k cores, diz-se que o grafo possui uma* ***k-coloração****. Um grafo G é k-colorível se existe uma k-coloração para G.*

**Definição 2.15** (Bondy, 1982 [2]). *Se $V_i(1 \leq i \leq k)$ é um conjunto de vértices em G, cada qual colorido com a cor i, então cada conjunto não-vazio $V_i$ é chamado de* ***classe de cor*** *de G.*

De acordo com [13], cada classe de cor $V_i$ representa um conjunto estável de $G$. O **grau de saturação** de um vértice $v \in V(G)$ é o número de vértices adjacentes a $v$ e previamente coloridos. O vértice $v_4$ da Figura 2.12 possui grau de saturação 3, pois é adjacente a três vértices coloridos de $G$.

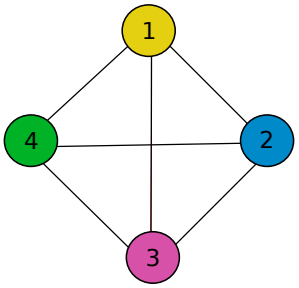



Figura 2.12: Exemplo de grafo colorido com 4 cores, ou seja, com uma 4-coloração.

**Definição 2.16.** *O* ***número cromático*** *de G é o mínimo inteiro positivo para o qual G é k-colorível e é denotado por $\chi(G)$ (lê-se chi de G).*

Na Figura 2.12, todos os vértices estão coloridos de acordo com as regras de coloração de vértices. Para a coloração do grafo $G$ foi usado o menor número de cores possível, ou seja, 4 cores. Dessa forma, tem-se que $\chi(G) = 4$.

Um grafo $G$ com número cromático igual a $k$ é um **grafo $k$-cromático**. Um grafo completo $G$ com $n$ vértices é $n$-cromático, uma vez que cada vértice de $G$ é adjacente a todos os $n-1$ vértices de $G$, serão necessárias $n$ cores para colorir $G$ [13].

Além da coloração de vértices, existem outras formas de coloração em grafos. Um exemplo é a **coloração de arestas**, na qual as cores são atribuídas às arestas de forma que arestas incidentes a um mesmo vértice não possuam as mesmas cores. Análogo à coloração de vértices, uma coloração de arestas com $k$ cores é chamada de **$k$-aresta-coloração**. Na Figura 2.13, o grafo $G$ da Figura 2.12 está com uma coloração de arestas.

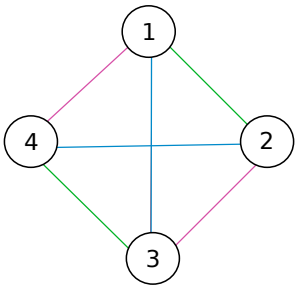



Figura 2.13: Coloração de arestas do grafo $G$ da Figura 2.12.

Uma outra forma de coloração é a **coloração total em grafos**, na qual os vértices e as arestas são coloridos. Neste tipo de coloração, valem as regras de coloração de vértices e de arestas, acrescentando que as arestas e seus respectivos vértices não podem compartilhar a mesma cor.

#### 2.1.3.1 Algumas variações do Problema de Coloração em Grafos

Ao estudar o Problema de Coloração em Grafos, verifica-se a existência de diversas variações do problema. Dentre tantas variações, um exemplo é conhecido como **$T$-coloração** de $G$. Neste tipo de coloração, é dado como um conjunto $T$ de inteiros não negativos com zero, onde uma coloração de $G$ é de forma que $|c(u) - c(v)| \notin T$ para todo $(u,v) \in E(G)$, ou seja, a distância entre duas cores de vértices adjacentes não deve pertencer ao dado conjunto $T$ [16].

Outra variação do problema de coloração é dada ao restringir as cores que podem ser usadas para colorir os vértices. Um exemplo desta variação é a **coloração com listas**, na qual cada vértice possui uma lista de cores possíveis que podem ser usadas na sua coloração. Ainda envolvendo coloração por listas, existe uma variação em que os vértices possuem lista de cores proibidas, ou seja, cores que não podem colorir o vértice. No Capítulo 3, o clássico problema de coloração por listas, conhecido como O Problema da Lista Coloração, será abordado com mais detalhes.

#### 2.1.3.2 Aplicações

O problema de coloração de vértices em grafos pode modelar diversos problemas do cotidiano. A modelagem destes problemas através do problema de coloração pode ajudar a encontrar uma solução para eles de forma mais rápida. Nesta seção, serão apresentadas algumas dessas aplicações.

- **Exemplo 1**. *Programação de Horário Escolar*. Um curso de uma universidade deseja oferecer *n* disciplinas em um semestre. Quando um aluno tenta efetuar matrícula em duas ou mais disciplinas que ocorram no mesmo horário, haverá um conflito de horários. O problema é: de que forma a distribuição de horários das disciplinas deve ser feita a fim de minimizar conflitos de horários? **Solução**: Este problema pode ser modelado da seguinte forma em coloração de grafos: os **vértices** serão as disciplinas, as **arestas** serão os conflitos de horários, o **número cromático** é o número mínimo de horários distintos (não conflitantes) e as **cores** serão disciplinas com menos conflito detectado na matrícula. Na Figura 2.14, um exemplo de grafo que modela o problema é mostrado. As disciplinas FCC (Fundamentos de Ciência da Computação) e OC (Organização de Computadores), assim como as disciplinas IA (Inteligência Artificial) e IHC (Interação Humano Computador), ocorrem no mesmo horário.

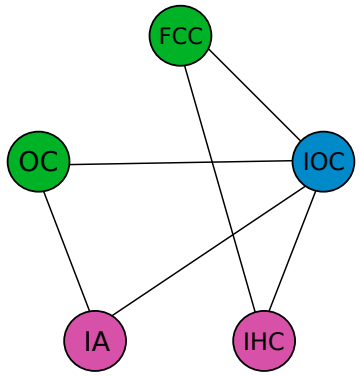
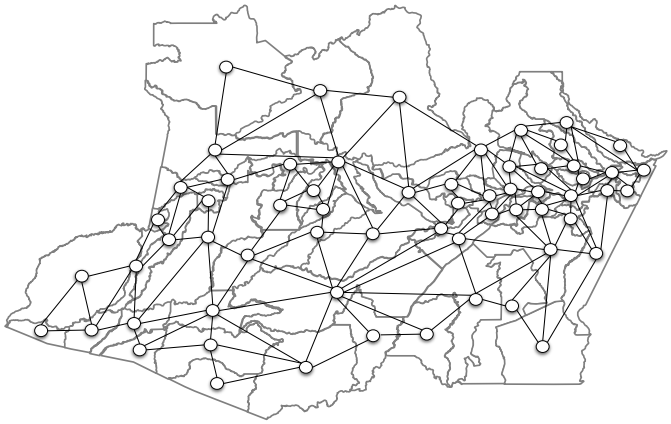


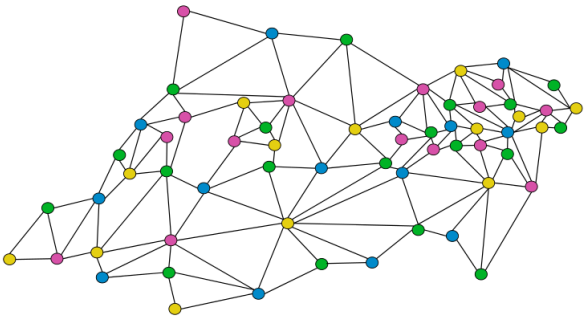

Figura 2.14: Exemplo 1 modelado em grafo. Os vértices são as disciplinas e as disciplinas com mesmas cores possuem conflito de horário.

- **Exemplo 2**. *Coloração de Mapas*. Uma gráfica deseja colorir um mapa do Amazonas de forma que as regiões vizinhas possuam cores diferentes para melhor visualização e que o gasto de cores seja o menor possível, já que para cada cor é necessário todo um trabalho de preparação do equipamento de impressão. O problema é: como colorir as regiões de maneira que o número de cores seja o menor possível para o mapa? **Solução**: Modelando o problema em coloração de grafos, temos que, os **vértices** são as cidades, as **arestas** representam a vizinhança das regiões e o **número cromático** representa o número mínimo de cores possíveis para colorir o grafo. Na Figura 2.15, o mapa do Amazonas é modelado em grafo e colorido seguindo as

regras de coloração em grafos.



(a)



(b)

Figura 2.15: Exemplo 2 modelado em grafo. Em (a), as cidades do estado do Amazonas são representadas como vértices do grafo $G$. Em (b), o grafo $G$ é colorido com 4 cores.

## 2.2 Teoria da Computação

A teoria da computação busca determinar os problemas que podem ser computados em um dado modelo de computação. Nesta seção, serão abordados conceitos básicos de otimização combinatória e complexidade computacional.

### 2.2.1 Otimização Combinatória

*Otimização combinatória* é um campo da matemática baseada no conjunto de técnicas de combinatória, programação matemática e teoria de desenvolvimento de algoritmos de resolução de problemas de natureza discreta. Segundo [9], um problema de otimização

consiste em procurar melhor solução possível para um determinado problema objetivando minimizar ou maximizar uma função objetivo.

**Definição 2.17.** *Uma **função objetivo** é uma função matemática que define a qualidade da solução em função das variáveis de decisão.*

**Definição 2.18.** ***Variáveis de decisão** são incógnitas a serem determinadas pela solução do modelo.*

A otimização combinatória possui problemas classificados em muitas maneiras possíveis. Alguns desses problemas apresentam métodos exatos para sua resolução. Outros necessitam de métodos heurísticos, visto que sua resolução exata seria de uma complexidade intratável. Dentre os problemas que apresentam métodos exatos, encontram-se os algoritmos em grafos, que serão abordados neste trabalho.

### 2.2.2 Complexidade Computacional

A teoria da complexidade computacional tem como objetivo classificar os problemas computacionais de acordo com sua dificuldade. Segundo [7], todo problema é tratado como sendo um conjunto de parâmetros que definem as instâncias e um conjunto de propriedades que configuram as restrições do problema a serem satisfeitas. Dessa forma, entre instâncias de um mesmo problema, as únicas variações estão nos conjuntos dos parâmetros.

Quando se trata do número de computações necessárias pare se obter a solução ótima, os problemas podem ser classificados em três grandes classes: P, NP e Intratáveis. De acordo com [7] temos que:

- **Problemas P** (*Polinomial Time*) são problemas que podem ser resolvidos em tempo polinomial. Isto significa que o esforço computacional cresce polinomialmente em função do tamanho da instância, além de que eles são resolvidos no tempo $O(n^k)$ sendo $k$ uma constante e $n$ o tamanho da entrada. Os problemas desta classe são também conhecidos como problemas tratáveis e existem algoritmos eficientes para resoluções de problemas desta classe.

- **Problemas NP** (*Nondeterminisc Polinomial Time*) são problemas "verificáveis"em tempo polinomial. Dada uma solução para o problema, pode-se verificar em tempo

polinomial se a solução satisfaz o problema de decisão. No entanto, o esforço computacional de encontrar uma solução cresce exponencialmente em função do tamanho da instância.

- **Problemas intratáveis** configuram problemas em que o esforço computacional cresce exponencialmente em função do tamanho da instância, com a garantia de que a solução encontrada seja a solução ótima para o problema.

Os problemas que estão na classe P podem ser resolvidos por uma técnica computacional conhecida por Força Bruta ou Técnica de Enumeração. Essa técnica resulta em um algoritmo de complexidade exponencial. As duas grandes classes de métodos de enumeração são: Enumeração Explícita e Enumeração Implícita.

- **Enumeração Explícita**: Conhecida também como busca exaustiva, método da força bruta, ou também como *Backtracking* sem podas ou cortes. Esse método gera todas as possíveis soluções de um problema, testando cada resposta com a restrição de achar a resposta correta. É um método ineficiente porque o número de possíveis soluções viáveis pode ser exponencial, mesmo para o caso de problemas em P. [12]

- **Enumeração Implícita**: Conhecida também como *Backtracking* Inteligente, ou seja, com cortes. Esse método fornece a melhor resposta sem precisar gerar todas as possíveis respostas, considerando tudo, mas sem 'gastar' o tempo de se calcular cada uma. Utiliza um conhecimento estrutural do problema, aliada a uma abordagem teórica mais aprofundada para propor uma estratégia computacional mais eficiente [12].

Dependendo do tamanho de entrada de algumas instâncias, as técnicas citadas consomem um tempo de execução não aceitável. Na maioria dos casos, é utilizado um algoritmo **heurístico** para isso. Segundo [12], o ideal seria elaborar um algoritmo de tempo de execução aceitável, além de que a solução encontrada seja uma solução ótima para o problema. O algoritmo heurístico geralmente não cumpre uma dessas propriedades. Isto se deve ao fato de que talvez seja um algoritmo que encontra boas soluções, mas que não possui a garantia de que sempre encontrará, ou possui um processamento rápido, mas não há certeza de que será rápido para todas as instâncias do problema [14]. Os algoritmos heurísticos são divididos em:

- **Heurísticas de Construção**, conhecidas também como **heurísticas gulosas**. Nestas heurísticas, o algoritmo constrói uma solução de acordo com algum critério de otimização (ou melhor elemento local), até que se tenha uma solução viável.

- **Heurísticas de Busca**, conhecidas também como **heurísticas de busca local**. Geralmente, elas começam trabalhando com alguma solução qualquer do problema, buscando melhorá-la através de operações de inserção, remoção ou troca até que não seja mais possível a melhoria da solução.

# Capítulo 3

# O Problema da Lista Coloração em Grafos

Neste capítulo será apresentado o Problema da Lista Coloração em Grafos, uma variação da coloração clássica de vértices com restrições adicionais em grafos, mais especificamente, com listas de cores atribuídas aos vértices. Serão apresentados os conceitos formais, caracterizações estruturais, os principais trabalhos existentes na literatura, bem como variações existentes de lista coloração e suas propriedades.

## 3.1 Definição e Conceitos Básicos

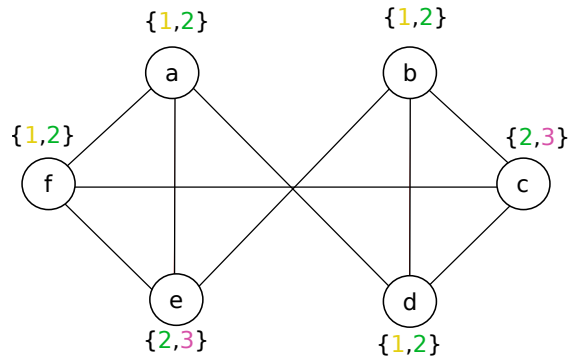
Dado um grafo $G$ para o qual existe um conjunto associado $L(v)$ de cores específicas para cada vértice $v$ de $G$. O conjunto $L(v)$ é chamado de **lista de cores** para $v$. Na Figura 3.1(a), um exemplo deste conceito é exibido.

**Definição 3.1** (Chartrand, 2008 [13])**.** *Uma **Lista Coloração** de G é uma coloração própria de G tal que $c(v) \in L(v)$ para cada $v \in V(G)$.*

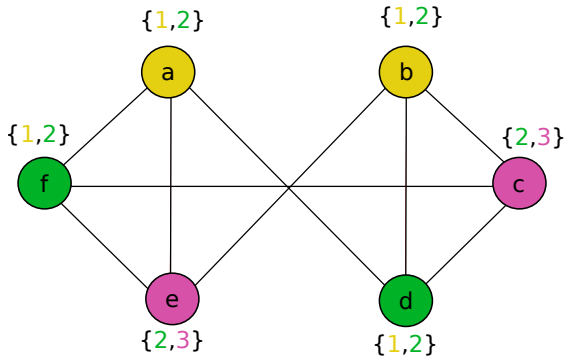
**Definição 3.2** (Chartrand, 2008 [13])**.** *Dada uma função de escolha $\mathcal{L}$, onde $\mathcal{L} = \{L(v) : v \in V(G)\}$. Se $\mathcal{L}$ é o conjunto de listas de cores para os vértices de $V(G)$ e existe uma lista coloração para este conjunto $\mathcal{L}$ de listas de cores, então G é dito ser $\mathcal{L}$**-lista-colorível**.*

A Figura 3.1(a) mostra um exemplo de grafo que possui listas de cores associadas aos seus vértices. Na Figura 3.1(b), é mostrada uma coloração dos vértices do grafo $G$ da

3.1(a), onde cada cor associada aos seus vértices pertence à sua lista de cores, logo este grafo é $\mathcal{L}$-lista-colorível.




(a)




(b)

Figura 3.1: Exemplo de uma lista coloração em grafo $G$. Em (a), grafo com listas de cores associadas a seus vértices. Em (b), grafo $G$ com uma coloração própria.

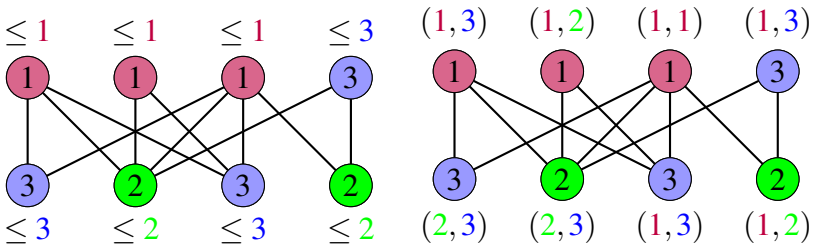
### 3.1.1 Variações de Lista Coloração

Existem algumas variações de lista coloração em grafos. A **$\mu$-coloração** e a **$(\gamma,\mu)$-coloração** são algumas delas. As definições de cada uma são dadas a seguir:

**Definição 3.3** (Bonomo, 2005 [3])**.** *Dado um grafo $G$ e uma função $\mu : V(G) \to \mathbb{N}$, $G$ é **$\mu$-colorível** se existe uma coloração $\mathcal{L}$ de $G$ tal que $f(v) \leq \mu(v)$ para todo $v \in V(G)$ .*

**Definição 3.4** (Bonomo, 2006 [4])**.** *Dado um grafo $G$ e uma função $\gamma,\mu : V(G) \to \mathbb{N}$ tal que $\gamma(v) \leq \mu(v)$ para todo $v \in V(G)$, $G$ é **$(\gamma,\mu)$-colorível** se existe uma coloração $\mathcal{L}$ de $G$ tal que $\gamma(v) \leq f(v) \leq \mu(v)$ para todo $v \in V(G)$.*

Um exemplo de **$\mu$-coloração** e de **$(\gamma,\mu)$-coloração** é apresentado na Figura 3.2.




(a) (b)

Figura 3.2: Exemplo de variações da lista coloração. Em (a), $\mu$-coloração de $G$. Em (b), $(\gamma,\mu)$-coloração de $G$.

### 3.1.2 Resultados sobre Lista Coloração encontrados na literatura

Na literatura, existem diversos resultados relacionados a lista coloração para várias classes de grafos. Segundo [18], o problema da lista coloração pode ser resolvido em tempo polinomial para árvores e grafos completos. Na Tabela 3.1, quase todas as classes de grafos apresentadas para lista coloração foram provadas por Jansen, em 1997 [18], exceto a lista coloração em grafos splits, provados por Alan Bertossi [1]. Já os resultados para $\mu$-coloração e $(\gamma,\mu)$-coloração foram provados mais recentemente por Bonomo [3] e [5].

Tabela 3.1: Tabela de Complexidade para Problemas de Lista Coloração e suas variações.

| **Classes de Grafos** | **Coloração** | ***k*-choosability** | **Referências** |
|---|---|---|---|
| **Grafos Bipartido** | Lista Coloração | NP-Completo | Jansen, 1997 [17] |
| | $\mu$-coloração | NP-Completo | Bonomo, 2005 [3] |
| | $(\gamma,\mu)$-coloração | NP-Completo | Bonomo, 2005 [3] |
| **Bipartido Completo** | Lista Coloração | NP-Completo | Jansen, 1997 [17] |
| | $\mu$-coloração | Polinomial | Bonomo, 2005 [3] |
| | $(\gamma,\mu)$-coloração | Polinomial | Bonomo, 2005 [3] |
| **Cografos** | Lista Coloração | NP-Completo | Jansen, 1997 [17] |
| | $\mu$-coloração | Polinomial | Bonomo, 2005 [3] |
| | $(\gamma,\mu)$-coloração | Aberto | - |
| **Grafos Split** | Lista Coloração | NP-Completo | Bertossi, 1999 [1] |
| | $\mu$-coloração | NP-Completo | Bonomo, 2009 [5] |
| | $(\gamma,\mu)$-coloração | Polinomial | Bonomo, 2009 [5] |

### 3.1.3 O Problema da Selecionabilidade

A lista coloração está fortemente ligada com **selecionabilidade em grafos**, que também foi introduzido por Paul Erdōs, Arthur Rubin e Herbert Taylor, em 1979 [11].

**Definição 3.5** (Chartrand, 2008 [13])**.** *Um grafo G é* ***k-selecionável*** *(ou k-choosable) se G*

*é* $\mathcal{L}$***-lista-colorível*** *(ou* $\mathcal{L}$***-choosable****) para toda coleção* $\mathcal{L}$ *de listas* $L(v)$ *para os vértices de* $G$ *tal que* $L(v) \leq k$ *para todo* $v \in V(G)$.

**Definição 3.6** (Chartrand, 2008 [13])**.** ***A lista número cromático*** *(do inglês, list chromatic number),* $\chi_\ell(G)$ *de* $G$ *é o mínimo inteiro positivo* $k$ *tal que* $G$ *é* $k$*-choosable.*

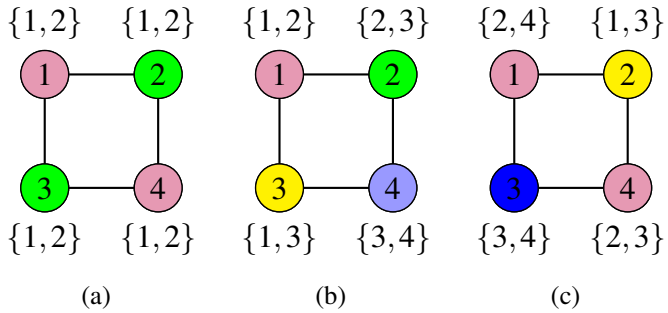



Figura 3.3: Exemplos de Grafo Ciclo que é 2-*choosable*.

O exemplo da Figura 3.3 mostra que um grafo ciclo par é 2-*choosable*, pois para **todas** as listas de tamanho 2, este grafo é $\mathcal{L}$**-lista-colorível**.

Existem alguns resultados com relação à *chosabilidade* em algumas classes de grafos. Em 2009, Pinar [23] mostrou que *k-choosable* é NP-Difícil para grafos livres de $P_5$. Mais recentemente, Juan [20], mostrou que a junção de dois caminhos $P_3$ e $P_n$ resulta em: $\chi_\ell(P_3 \vee P_m) = \chi_\ell(P_3) + \chi_\ell(P_m)$, se $1 \leq n \leq 15$.

A classe de grafos planares têm sido um dos objetos de estudos principais em selecionabilidade. De acordo com o Teorema das Quatro Cores, o número cromático de um grafo planar é no máximo 4 [10]. Em 1979, Erdös, Rubin e Taylor conjecturaram que a lista número cromático de um grafo planar é no máximo 5 [11]. Em 1994, Thomassen [25] completou a verificação dessa conjectura.

Em 1993, Margit Voigt [26] apresentou um exemplo de um grafo planar de ordem 238, que não é 4-*choosable*. Em 1996 Maryam Mirzakhani [21] deu um exemplo de um grafo planar de ordem 63, que não é 4-*choosable*. Alon e Tarsi [15] mostraram que todo grafo bipartido e planar é 3-*choosable*.

# Capítulo 4

# Resultados Alcançados

Neste capítulo, serão explorados os principais resultados obtidos com o trabalho. Aqui, serão mostrados os algoritmos implementados durante o desenvolvimento do projeto para o Problema da Lista Coloração.

Dentre os algoritmos implementados, estão: algoritmo de reconhecimento de um grafo lista-colorido; um algoritmo de enumeração explícita para verificar se um grafo é $\mathcal{L}$-lista-colorível (ou $\mathcal{L}$-*choosable*) para uma lista de cores associadas aos seus vértices, implementado junto à aluna de mestrado Simone Gama; algoritmos gulosos de coloração arbitrária, tanto para coloração simples quanto para lista-coloração; e algoritmos baseados na heurística DSATUR para coloração clássica de vértices e para lista-coloração.

## 4.1 Algoritmo de Reconhecimento se um Grafo está Lista-Colorido

O Algoritmo 1 exibe o pseudocódigo do algoritmo implementado para o reconhecimento de um grafo lista-colorido. Ele recebe um grafo $G$ e verifica se $G$ está colorido seguindo as regras do problema da lista coloração, ou seja, se os vértices estão coloridos com cores de suas listas de cores e se os vértices adjacentes possuem cores diferentes.

**Algoritmo 1:** Está_Lista_Colorido($G = (V, E)$)

```
1  início
2      para cada v ∈ G faça
3          se CorVertice(v) ∈ L(v) e nenhum vértice adjacente de v está colorido com
           CorVertice(v) então
4              retorne SIM;
5          fim se
6          senão
7              retorne NAO;
8          fim se
9      fim para cada
10 fim
```

## 4.2 Algoritmo de Reconhecimento se um Grafo é Lista-Colorível

O algoritmo para reconhecer se um grafo é lista-colorível foi implementado utilizando o método de enumeração explícita. O algoritmo verifica se existe uma coloração própria do grafo com as listas de cores associadas aos vértices. Quando o algoritmo encontra uma coloração própria para o grafo, o restante da árvore de enumeração de possíveis soluções não precisa ser visitado. No Algoritmo 2 está descrito o pseudocódigo criado.

O algoritmo inicia com uma condição de parada *falso*, controlada pelo valor *booleano* da variável *existe*, na linha 1. No início da execução, o grafo ainda não está colorido, então o algoritmo segue para a linha 7 para o conjunto de listas *L*. Após chegar na linha 9, o vértice é colorido com as cores de sua respectiva lista de cores e o algoritmo chama recursivamente a função *Existe_Lista_Coloracao* para colorir o vértice $i+1$ até que todos os vértices estejam coloridos. Dessa forma, se existe uma coloração para o grafo com a lista de cores dada, o algoritmo termina sua execução (linha 12).

**Algoritmo 2:** Existe_Lista_Coloracao($G, L, existe, i$)

```
1  se !(existe) então
2      se L = ∅ então
3          se ExisteColoracao(G) então
4              existe ⟵ SIM;
5          fim se
6      senão
7          l ⟵ remove o primeiro da lista L;
8          para cada cor ∈ l faça
9              G.v_i.cor ⟵ cor;
10             Existe_Lista_Coloracao(G, L, existe, i + 1);
11             se (existe) então
12                 break;
13             fim se
14         fim para cada
15     fim se
16 fim se
```

Se todos os vértices estão coloridos, mas não existe uma coloração própria para o grafo, o algoritmo segue para a próxima cor da lista de cores do último vértice visitado. Se ainda assim não existir coloração própria e não houver mais cores disponíveis na lista de cores à serem testados, o algoritmo volta um nível na árvore de execução para testar a próxima cor disponível desse vértice, e assim por diante. O processo é repetido até que se encontre uma coloração válida para o conjunto de listas de cores associadas ao grafo.

## 4.3 Algoritmo Guloso de Coloração: Ordem Decrescente de Grau

O Algoritmo 3 exibe o pseudocódigo do algoritmo implementado para colorir um grafo $G$ levando em conta o grau dos vértices, sendo, portanto, um algoritmo de abordagem gulosa.

Na linha 2, o algoritmo ordena os vértices por ordem decrescente de grau. Após isto,

na linha 3, o algoritmo atribui a cor 1 ao vértice que possui o maior grau. Ao chegar na estrutura de repetição da linha 4, o algoritmo seleciona cada um dos vértices de $G$ de acordo com a ordenação decrescente do grau. Depois de selecionar um vértice, na linha 6 é atribuída a menor cor disponível ao vértice. O processo é repetido até que todo o grafo seja colorido.

**Algoritmo 3:** Colorir_OrdemDecrescenteGrau($G = (V, E)$)

```
1 início
2     Ordena os vértices V(G) em ordem decrescente de grau;
3     O vértice de maior grau recebe a cor 1;
4     enquanto ∃ vértice descolorido faça
5         Seleciona o vértice da vez na ordem decrescente de grau;
6         Colore o vértice com a menor cor disponível;
7     fim enquanto
8 fim
```

Análogo ao Algoritmo 3, foi implementado um algoritmo para lista-colorir um grafo de acordo com a ordem decrescente de grau. O pseudocódigo desta implementação é exibido no Algoritmo 4.

**Algoritmo 4:** L_Colorir_OrdemDecrescenteGrau($G = (V, E)$)

```
1 início
2     Ordena os vértices V(G) em ordem decrescente de grau;
3     O vértice v de maior grau recebe uma das cores contidas na sua lista de cores
      L(v);
4     enquanto ∃ vértice descolorido faça
5         Seleciona o vértice da vez na ordem decrescente de grau;
6         Colore o vértice com uma das cores de sua lista L(v), de maneira que essa
          cor não conflita com a cor de seu vértice adjacente;
7     fim enquanto
8 fim
```

Inicialmente, na linha 2, o algoritmo ordena os vértices do grafo $G$ por ordem decrescente de grau. Na linha 3, o vértice $v$ de maior grau é colorido com uma das cores

pertencentes à sua lista de cores $L(v)$. Enquanto houver um vértice do grafo que esteja descolorido, o vértice de maior grau é selecionado e, na linha 6, ele é colorido com uma cor pertencente a $L(v)$. Após colorir todos os vértices, o grafo estará colorido com uma lista-coloração própria.

## 4.4 Algoritmo baseado na Heurística DSATUR para Lista Coloração

O algoritmo DSATUR (sigla para *Degree of Saturation*) é um algoritmo heurístico que utiliza uma abordagem gulosa, levando em conta o grau de saturação dos vértices. Ele foi proposto por Daniel Brélaz em 1979 [6]. A seguir, estão descritos os passos feitos pelo DSATUR para a coloração de um grafo:

1. Primeiramente, o algoritmo ordena os vértices por ordem decrescente de grau;
2. O vértice de maior grau é inicialmente colorido com a cor 1. Neste momento, o grau de saturação dos vértices vizinhos ao vértice colorido é acrescido de um;
3. Na iteração seguinte, o vértice de maior grau de saturação é selecionado. Caso existam vértices com o mesmo grau de saturação, escolhe-se o vértice de maior grau. Persistindo o empate, seleciona-se o vértice de menor índice. Por fim, o vértice escolhido será colorido com a menor cor disponível, a fim de que o número de cores não aumente muito e de modo que essa cor não conflite com seus vértices adjacentes;
4. Repete-se a iteração até que todos os vértices fiquem coloridos adequadamente.

No Algoritmo 5 está descrito o pseudocódigo do DSATUR para a coloração clássica de vértices. O algoritmo recebe como entrada um grafo $G$ e, na linha 2, inicia a ordenação dos vértices por ordem decrescente de grau. Após isto, na linha 2, atribui-se a cor 1 ao vértice de maior grau. Nas linhas 6-8, o algoritmo procura o vértice de maior grau de saturação. Caso existam dois ou mais vértices com o maior grau de saturação, o algoritmo escolhe o vértice de maior grau. Após a escolha do vértice, ele finalmente é colorida na linha 9. O processo se repete até que o grafo seja totalmente colorido.

**Algoritmo 5:** DSATUR($G = (V, E)$)

```
1  início
2      Ordena os vértices V(G) em ordem decrescente de grau;
3      O vértice de maior grau recebe a cor 1;
4      enquanto ∃ vértice descolorido faça
5          v ← grau_max_saturacao(G);
6          se ∃ mais de um vértice com grau de saturação igual a v então
7              Escolhe o vértice de maior grau;
8          fim se
9          Colore o vértice com a menor cor disponível;
10     fim enquanto
11 fim
```

A estratégia do algoritmo DSATUR foi utilizada na implementação de um algoritmo para colorir um grafo baseado no problema de coloração por listas. O pseudocódigo é mostrado no Algoritmo 6. Neste algoritmo, em vez de utilizar a menor cor disponível no momento de colorir, serão utilizadas as cores disponíveis na lista de cores de cada vértice $v \in V(G)$.

No Algoritmo 6, o processo de coloração é semelhante ao DSATUR original, porém, as cores utilizadas para colorir um vértice $v$ pertencem à lista de cores de $v$. Tal processo pode ser visto na linha 3, na qual o vértice de maior grau recebe uma cor de sua lista de cores. Da mesma forma acontece no momento de colorir o vértice com maior grau de saturação, ou seja, a cor escolhida para colorir $v$ pertence à lista de cores desse vértice, sem conflitar com a cor de um vértice adjacente.

Visto que o algoritmo possui característica gulosa, ele não funciona em alguns casos de grafos. Na Figura 4.1, estão representados dois exemplos de coloração através do DSATUR para lista-coloração. Quando o algoritmo tenta colorir o vértice $e$ da Figura 4.1(a), as cores disponíveis em sua lista de cores já foram utilizadas nos vértices $a$ e $b$, não sendo possível colorir o grafo. No exemplo da Figura 4.1(b), é possível obter uma coloração própria para o grafo com as listas de cores disponíveis.

**Algoritmo 6:** L-DSATUR($G = (V, E)$)

**início**
  Ordena os vértices $V(G)$ em ordem decrescente de grau;
  O vértice $v$ de maior grau recebe a cor $c \in L(v)$;
  **enquanto** $\exists$ *vértice descolorido* **faça**
    $v \leftarrow grau\_max\_saturacao(G)$;
    **se** $\exists$ *mais de um vértice com grau de saturação igual* **então**
      *Escolhe o vértice de maior grau*;
    **fim se**
    *Colore o vértice $v_i$ com uma cor de sua lista de cores $L(v)$, de maneira que essa cor não conflita com a cor de seu vértice adjacente;*
  **fim enquanto**
**fim**

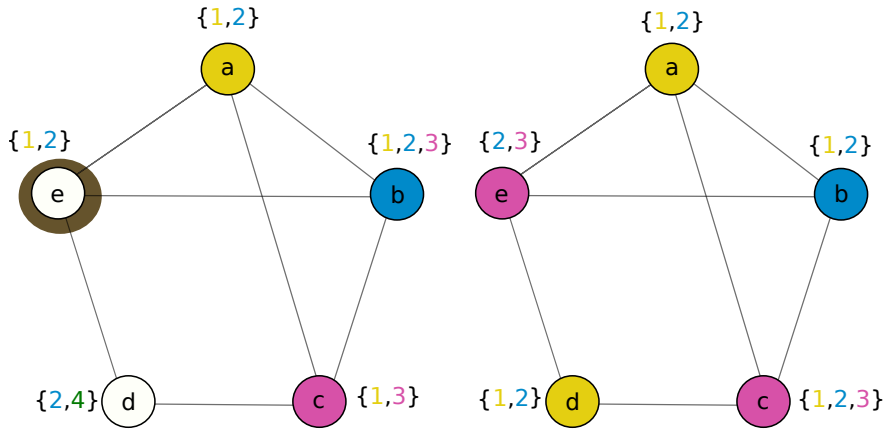



Figura 4.1: Exemplo de execução do Algoritmo L-DSATUR. Em (a), um exemplo de um grafo com lista de cores onde o L-DSATUR não funciona. Em (b), um exemplo onde L-DSATUR funciona.

# Capítulo 5

# Considerações Finais

Este trabalho abordou o estudo do Problema da Lista Coloração em Grafos. Para isto, fez-se necessário o estudo de algumas determinadas áreas da computação, como Teoria dos Grafos e Complexidade Computacional. O estudo foi feito desde os conceitos básicos de coloração em grafos e lista coloração a propriedades e funcionamento de algoritmos. Além disso, foram expostas as definições destes conceitos, bem como as principais abordagens teóricas e resultados alcançados.

## 5.1 Contribuições do PIBIC

O PIBIC 2014/2015 cumpriu com o objetivo de proporcionar uma iniciação científica em nível de graduação e aumentar o interesse da aluna pela área estudada. A partir de palestras e revisões da literatura, a autora deste trabalho adquiriu grande conhecimento sobre os problemas e variações de coloração em grafos, bem como consolidou seu interesse pela área de Teoria dos Grafos e Otimização Combinatória. Além disso, participando do PIBIC, a aluna teve a oportunidade de participar do Congresso da Sociedade Brasileira de Computação (CSBC), realizado em Julho de 2015. Dessa forma, ela pôde adquirir conhecimento sobre diferentes áreas apresentadas em cursos no CSBC, mas focando sempre em conhecer mais sobre a área que ingressou e fez projeto de iniciação científica. A aluna está muito mais motivada com as oportunidades que surgirão, como novos projetos e futuros congressos e, principalmente, com o que pode aprender nos próximos anos.

## 5.2 Trabalhos Futuros

Devido aos bons resultados no trabalho do PIBIC 2014/2015 e ao enorme interesse da aluna em continuar na área, um novo trabalho surgiu com o seguinte tema: Algoritmos Online e Teoria dos Jogos na Otimização de Sistemas Dinâmicos com Conflitos. Ao longo desses 12 meses de projeto, a aluna aproximou-se da área com a orientação da professora Rosiane de Freitas. Sendo assim, a professora a convidou para participar de mais um PIBIC em nível de graduação, mesmo que a aluna ainda seja nova no curso de Computação. O estudo, as pesquisas e todo conhecimento adquirido neste atual projeto serão de grande importância para o trabalho futuro, visto que muitos dos conceitos relacionados à Teoria de Grafos, Otimização Combinatória e à Complexidade Computacional já foram bem estudados e serão aprofundados.

# Referências


[1] BERTOSSI, A. L(2,1,1)-labeling problem on graph. *Discrete Applied Mathematics* (1999).

[2] BONDY, J. A., AND MURTY, U. S. R. *Graph Theory with Applications*. Elsevier Science Publishing, 1982.

[3] BONOMO, F., AND CECOWSKI, M. Between coloring and list-coloring: $\mu$-coloring. *Electronic Notes in Discrete Mathematics 19* (2005), 117–123.

[4] BONOMO, F., DURÁN, G., AND MARENCO, J. Exploring the complexity boundary between coloring and list-coloring. *Electronic Notes in Discrete Mathematics 25* (2006), 41–47.

[5] BONOMO, F., AND MARENCO, J. Exploring the complexity boundary between coloring and list-coloring. *Eletronic Notes in Discrete Mathematics* (2009).

[6] BRÉLAZ, D. New methods to color the vertices of a graph. *Communications of the ACM* (1979).

[7] CORMEN, T. H., LEISERSON, CHARLES E.AND RIVEST, R. L., AND STEIN, C. *Algoritmos - Teoria e Prática*, 2th ed. Editora Campus, 2002.

[8] COSTA, P. P. D. Teoria de grafos e suas aplicações. Mestrado profissional em matemática universitária, Campus de Rio Claro, 2011.

[9] E KENNETH STEIGLITZ, C. H. P. *Combinatorial Optimization: Algorithms and Complexity*. Dover Books on Computer Science, 1998.

[10] E W.HAKEN, K. A. Every planar map is four colourable, part i. *Discharging* (1995).

[11] ERDOS, P., AND RUBIN, L. Chosability in graphs. *Proceedings, West Coast Conference on Combinatorics* (1979).

[12] GAMA, S. Sobre o problema da lista coloração em grafos. Mestrado, Universidade Federal do Amazonas, UFAM, 2014.

[13] GARY CHARTRAND, P. Z. *Chromatic Graph Theory*. CRC Press, 2008.

[14] GOLDBARG, MARCO C. E PACCA, H., AND LUNA, L. *Otimização Combinatória e Programação Linear*, 2th ed. Editora Campus, 2005.

[15] GUTNER, S., AND TARSI, M. Some results on (a : b)-choosability. *Discrete Mathematics* (2009).

[16] HALE, W. K. Frequency assignment: Theory and applications. *Proceedings of the IEEE* (1980).

[17] JANSEN, K. The optimum cost chromatic partition problem. *Lecture Notes in Computer Science* (1997).

[18] K. JANSEN, P. S. Generalized colorings for tree-like graphs. *Discrete Applied Mathematics* (1997).

[19] LE, V. B., BRANDSTĀDT, AND SPINRAD, J. *Graph Classes: A Survey*. 1999.

[20] LIU, H.-L., AND JUAN, J. S.-T. List coloring of join of paths. *Whorkshop Combinatoria* (2001).

[21] MIRZAKHANI, M. A small non-4-choosable planar graph. *Bull. Inst. Combin. Appl* (1996).

[22] ORE, O. *Graphs and their Uses*, 2th ed. Washington: The Mathematical Association of America, 1990.

[23] PETR A. GOLOVACH AND PINAR HEGGERNES. Choosability of $p_5$-free graphs. *Department of Informatics, University of Bergen* (1999).

[24] SZWARCFITER, J. L. *Grafos e Algoritmos Computacionais*. Elsevier, 1986.

[25] THOMASSEN, C. Every planar graph is 5-choosable. *Journal of Combinatorial Theory* (1994).

[26] VOIGT, M. List colourings of planar graphs. *Elsevier Science Publishers* (1993).