UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE ESTATÍSTICA E MATEMÁTICA APLICADA
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM E MÉTODOS QUANTITATIVOS

VICTOR LAGE PESSOA

ESTUDO DE COMPLEXIDADE DE JOGOS DE COLORAÇÃO

FORTALEZA
2019

VICTOR LAGE PESSOA

ESTUDO DE COMPLEXIDADE DE JOGOS DE COLORAÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem e Métodos Quantitativos da Universidade Federal do Ceará (UFC) como requisito parcial para a obtenção do Título de Mestre em Modelagem e Métodos Quantitativos. Área de Concentração: Otimização e Inteligência Computacional.

Orientador: Prof. Dr. Ronan Soares

Co-Orientador: Prof. Dr. Rudini Sampaio

FORTALEZA
2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

P568e Pessoa, Victor Lage.
Estudo de Complexidade de Jogos de Coloração / Victor Lage Pessoa. – 2019.
66 f. : il.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Modelagem e Métodos Quantitativos, Fortaleza, 2019.
Orientação: Prof. Dr. Ronan Pardo Soares.
Coorientação: Prof. Dr. Rudini Menezes Sampaio.

1. Teoria dos Grafos. 2. Complexidade Computacional. 3. Jogos de Coloração. I. Título.
CDD 510

VICTOR LAGE PESSOA

ESTUDO DE COMPLEXIDADE DE JOGOS DE COLORAÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem e Métodos Quantitativos da Universidade Federal do Ceará (UFC) como requisito parcial para a obtenção do Título de Mestre em Modelagem e Métodos Quantitativos. Área de Concentração: Otimização e Inteligência Computacional.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Ronan Soares (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Rudini Soares (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Carlos Diego Rodrigues
Universidade Federal do Ceará (UFC)

Prof. Dr. Fabrício Benevides
Universidade Federal do Ceará (UFC)

À todos que não desistem, mesmo quando tudo parece estar perdido.

AGRADECIMENTOS

Eu não poderia deixar de agradecer algumas pessoas importantes que me ajudaram ao longo desses dois maravilhosos anos de mestrado.

Agradeço primeiramente ao professor Ronan Soares pela oportunidade de ser seu primeiro orientando de mestrado, espero não ter dado tanto trabalho e que continuemos amigos ao longo dos anos. De forma análoga, agradeço o professor Rudini Sampaio por aceitar ser meu co-orientador e por sua atitude bem humorada em todas reuniões que tivemos.

Ao professor Ricardo Coelho por sua dedicação ao programa de mestrado e ter conseguido uma bolsa de estudos para mim. Tenho certeza que não teria conseguido chegar onde cheguei sem sua ajuda, sou bastante grato.

À professora Cláudia Linhares por ter sido a pessoa que me introduziu à teoria dos grafos de forma aprofundada e bem didática. Sua disciplina me mostrou que até pontinhos e linhas podem ser bem desafiadoras.

Agradeço aos membros da banca Ana Karolinna e Carlos Diego pelas correções e críticas que deixaram esta dissertação o mais profissional possível.

Um agradecimento especial à todos os membros do ParGO por serem consistentemente os pesquisadores mais engraçados e animados do departamento, a energia e paixão que vocês proporcionam me motivam a ser um pesquisador e colega de laboratório melhor.

Aos meus amigos próximos, Luiz Alberto, André Novais, Pedro Gustavo, Alejandro Yugar e Felipe Gomes por sempre no final do dia ter alguém para conversar e desabafar em tempos difíceis.

Por fim, à minha noiva Ana Carolina Batista por ser a pessoa que irei amar pelo resto dos meus dias, tudo que faço é para deixar ela orgulhosa de mim.

“Esperança é uma boa coisa, talvez a melhor das coisas, e nenhuma coisa que é boa jamais irá morrer” (ANDY DUFRESNE)

RESUMO

Respondemos nesta dissertação uma questão que permaneceu em aberto nos últimos 28 anos: a complexidade do jogo de coloração em grafos de Bodlaender. O jogo de coloração em um grafo G simples é jogado por duas pessoas: Alice e Bob, cada um dispõe do mesmo conjunto de cores C com k cores distintas. Em turnos alternados, cada um escolhe apenas um vértice ainda não colorido de G e uma cor de C usada para colorir tal vértice de forma que vértices vizinhos possuam cores distintas. Alice ganha o jogo quando todos so vértices de G são propriamente coloridos e Bob ganha se em algum momento existir um vértice que não possa ser colorido com nenhuma das k cores de C. Estudamos também a complexidade de uma variante do jogo chamada de jogo de coloração gulosa e obtemos o número de Grundy lúdico para grafos da classe P_4-esparso em tal jogo.

Palavras-chaves: Coloração em grafos. Número Cromático Lúdico. Número de Grundy Lúdico.

ABSTRACT

We answer in this dissertation a question that remained unanswered for the last 28 years: the complexity of Bodlaender's graph coloring game. The coloring game in a simple graph G is played by two players: Alice and Bob, each has the same set of colors C having k distinct colors. In alternating turns, each chooses a single not yet colored vertex from G and a color from C used to color such vertex in a way that adjacent vertices have distinct colors. Alice wins the game when all vertices from G are properly colored and Bob wins if at any moment there is a vertex which cannot be colored by any of the k colors from C. We also studied the complexity of a variant of the game called the greed coloring game and obtained the game Grundy number for graphs of the P_4-sparse class in such game.

Key-words: Graph coloring. Game Chromatic Number. Game Grundy Number.

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

A estrutura conhecida como grafo é objeto de estudo presente em vários trabalhos na literatura, como por exemplo nas áreas de ciência da computação e otimização combinatória. O propósito da estrutura é estudar relações que objetos possuem entre si. Um objeto pode representar qualquer entidade que possua relações com outras entidades, podemos citar como exemplo os países de um mapa mundi e a relação de vizinhança entre os países.

Um dos mais famosos problemas relacionados à teoria dos grafos é o chamado "Problema das quatro cores". O problema consiste em mostrar que não mais do que quatro cores distintas são necessárias para colorir as regiões de um plano de tal forma que regiões que compartilhem uma fronteira não possuam a mesma cor. O problema das quatro cores foi umas das principais motivações para o estudo e aprofundamento da área de coloração em grafos.

O problema foi resolvido com a assistência de computadores que foram usados para analisar uma grande quantidade de casos particulares. Apesar de provada, parte da comunidade de teoria dos grafos não aceitou a prova como definitiva. Alguns pesquisadores se perguntaram se não haveria uma forma mais simples de resolver o problema, se possível sem o auxílio de máquinas.

Foi então que Bodlaender em (BODLAENDER, 1991) propôs abordar o problema como um jogo na esperança de obter uma prova sem o uso de máquinas. Dois jogadores tomariam turnos alternados colorindo as regiões do plano usando uma quantidade predeterminada de cores disponíveis, respeitando a regra de que regiões vizinhas não tivessem a mesma cor. Em uma das versões do jogo os competidores tem objetivos distintos, uma competidora, Alice, tenta colorir completamente o plano com as cores disponíveis enquanto o outro, Bob, tenta impedir que isso aconteça, isto é, forçar uma região ainda não colorida a possuir regiões vizinhas com todas as cores disponíveis, impossibilitando que ela possa ser colorida. O jogo descrito acima foi chamado de "jogo da construção de coloração".

No contexto de grafos, chamamos tais regiões de *vértices* e se um vértice é vizinho de outro dizemos que existe uma *aresta* que os liga. A coloração de regiões proposta acima é chamada na literatura de coloração *própria*. Um parâmetro que todo grafo possui é o seu *número cromático* que é a quantidade de cores necessária para que o grafo em questão

possua uma coloração própria.

O problema de determinar o número cromático de um grafo é bastante difícil. Existem problemas considerados "fáceis" onde determinar uma solução para o problema pode ser feita, no pior caso, em tempo polinomial e existem problemas "difíceis" que são os chamados de problemas NP-Completos (mais sobre problemas NP-Completos na Seção 2 deste texto). Determinar o número cromático de um grafo é um problema NP-Completo.

Um parâmetro similar ao número cromático foi introduzido junto com o jogo de coloração que é o chamado *número cromático lúdico*. O número cromático lúdico é definido como a quantidade mínima de cores que Alice necessita para garantir a vitória no jogo de construção de coloração.

Além do jogo de construção de coloração ainda existem variantes que adicionam regras ligeiramente diferentes, como é o caso do jogo de coloração gulosa, proposto por Frederic Havet em (HAVET; ZHU, 2013). Além das regras citadas anteriormente, os jogadores devem sempre usar a menor cor disponível para fazer a sua coloração, uma cor no caso sendo um número natural e a menor cor disponível sendo o menor número natural que não aparece na vizinhança de um vértice não colorido.

O principal objetivo desta dissertação é determinar a complexidade do jogo de coloração, que é um problema que está em aberto a mais de 28 anos, e do jogo de coloração gulosa.

Os resultados dos capítulos 4 e 5 foram enviados em forma de resumo estendido e aceitos para o Simpósio Latino Americano de Algoritmos, Grafos e Otimização de 2019 (LAGOS 2019).

1.2 Organização do texto

No Capítulo 2 expomos definições e notações relevantes para a melhor compreensão desta dissertação.

A descrição detalhada e uma revisão bibliográfica do problema a ser tratado se encontram no Capítulo 3 com uma recapitulação, não necessariamente em ordem cronológica, dos principais resultados sobre o jogo de coloração.

Temos no Capítulo 4 uma análise de complexidade para o jogo de coloração em suas versões clássica e gulosa.

Para o Capítulo 5, estudamos o jogo de coloração gulosa para uma classe de grafos chamada de P_4-laden estendido e mostramos valores exatos para a quantidade mínima de cores que Alice necessita para garantir uma estratégia vencedora para o jogo de coloração

gulosa.

Por fim, no Capítulo 6 temos as considerações finais dos resultados obtidos nesta dissertação, assim como sugestões para trabalhos futuros.

2 NOTAÇÕES E DEFINIÇÕES

Neste capítulo, são definidos conceitos e notações que usamos ao longo da dissertação. Para melhor compreensão do jogo de coloração, e sua complexidade, é necessário à priori definir formalmente conceitos de teoria dos grafos e complexidade computacional.

2.1 Teoria dos Grafos

2.1.1 Conceitos Iniciais

Um grafo G é um par ordenado (V, E) onde V é chamado o conjunto não vazio de *vértices* de G e E é o conjunto de *arestas* de G. Os conjuntos V e E são disjuntos. A *função de incidência* Ψ_G associa cada aresta a um conjunto não vazio e de no máximo 2 vértices de G. Sendo e uma aresta e $\Psi_G(e) = \{u, v\}$ (ou somente uv), dizemos que u e v são as *extremidades* da aresta e. Adicionalmente, dizemos que se u e v são extremidades de uma aresta e, então u e v são *vizinhos* e se um vértice v qualquer é extremidade de uma aresta e qualquer, então v é *incidente* a e e vice versa. Para um vértice v qualquer em um grafo G, denotamos por $N_G(v)$ o conjunto de vizinhos de v em G e por $N_G[v]$ o conjunto $N_G(v) \cup \{v\}$. chamamos $N_G(v)$ e $N_G[v]$ de *vizinhança aberta* (ou apenas vizinhança) e *vizinhança fechada* de v, respectivamente.

Podemos estender a definição de vizinhança aberta e a vizinha fechada para um subconjunto de vértices de G. Seja $V' \subseteq V$ um subconjunto de vértices de um grafo G, a vizinhança aberta $N_G(V')$ é definida como o conjunto de vértices de V que são vizinhos de vértices de V'. Por $N_G[V']$ denotamos a vizinhança fechada do subconjunto de vértices V' que representa o subconjunto de vértice de G que são vizinhos de vértices de V' ou pertencem à V'.

Para um vértice v de um grafo G, denotamos por $d(v)$ o *grau* de v, ou a quantidade de vizinhos que v possui. Denotamos por $\Delta(G)$ o *grau máximo* de G, isto é, a quantidade de vizinhos que o vértice com mais vizinhos em G possui (podendo existir mais de um vértice com $\Delta(G)$ vizinhos).

Podemos graficamente representar um grafo como um conjunto objetos e segmentos de reta (ou curvas) que ligam tais objetos, estes representando respectivamente os vértices e arestas do grafo. Na Figura 2.1 temos como exemplo um grafo $G = (V, E)$ com um conjunto

de vértices $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, um conjunto de arestas $E = \{e_1, e_2, e_3, e_4, e_5\}$ e a função de incidência Ψ_G definida por

$$\Psi_G(e_1) = v_1v_2, \quad \Psi_G(e_2) = v_2v_3, \quad \Psi_G(e_3) = v_3v_4, \quad \Psi_G(e_4) = v_5v_2, \quad \Psi_G(e_5) = v_3v_6.$$

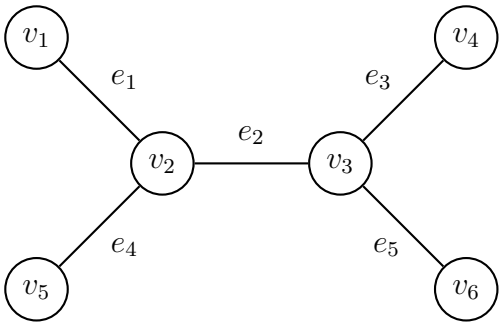


Figura 2.1: Ilustração de um grafo $G = (V, E)$

Quando duas arestas distintas possuem as mesmas extremidades, dizemos que essas arestas são *paralelas*. Uma aresta que possui as duas extremidades iguais é chamada de *loop*. Para melhor visualizar os dois tipos de arestas mencionadas acima, temos como exemplos as arestas paralelas e_2 e e_5 da Figura 2.2 e o loop e_3 na mesma figura. Um grafo que não possui arestas paralelas ou loops, isto é, sua função de incidência ψ_G é injetiva, associando cada aresta a um conjunto de dois vértices distintos é chamado de grafo *simples*.

A partir deste ponto da dissertação, quando estivermos nos referindo a qualquer grafo G, este grafo é simples e estaremos definindo seu conjunto de arestas como pares não ordenados de vértices de G, onde uma aresta é representado simbolicamente como uma concatenação os vértices que a compõem (se uma aresta possui extremidades u e v, então ela é representada como uv, por exemplo).

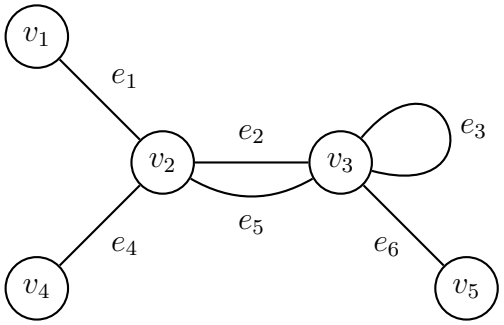


Figura 2.2: Ilustração de um grafo $G = (V, E)$ com arestas paralelas e loops.

A *ordem* de um grafo G é definida como a quantidade de vértices em G e o *tamanho* de G é definido como a quantidade de arestas em G.

Dizemos que um grafo $\overline{G} = \{\overline{V}, \overline{E}\}$ é *complementar* de um grafo G se $\overline{V} = V$ e uv pertence à $\overline{E}$ se, e somente se, uv não pertence à E.

Podemos fazer operações de deleção, isto é, remoção de arestas e vértices em qualquer grafo $G = (V, E)$ para obter um grafo com tamanho menor que G. A operação de *remoção de aresta* consiste em remover uma aresta uv de E e manter suas extremidades em V, enquanto a *remoção de vértice* consiste em remover um vértice v de V assim como todas as arestas E que possuam uma extremidade em v.

Um grafo $G' = (V', E')$ é um *subgrafo* de um grafo $G = (V, E)$ se $V' \subseteq V$ e $E' \subseteq E$. Um grafo G' é um subgrafo *induzido* de um grafo G se puder ser obtido a partir de G através de apenas remoções de vértices. Denotamos por $G[V']$ o subgrafo induzido de G cujo conjunto de vértices é V'.

Dois grafos, digamos G e H, são *isomorfos* se existem bijeções $\theta : V(G) \to V(H)$ e $\phi : E(G) \to E(H)$ tais que $\Psi_G(e) = uv$ se e somente se $\Psi_H(\phi(e)) = \theta(u)\theta(v)$. O par de mapeamento (θ, ϕ) é chamado de *isomorfismo* entre G e H. Podemos tomar como exemplo os grafos da Figura 2.3 onde temos um par de mapeamentos

$$\theta := \begin{pmatrix} v_1 & v_2 & v_3 & v_4 \\ u_1 & u_2 & u_3 & u_4 \end{pmatrix}, \quad \phi := \begin{pmatrix} e_1 & e_2 & e_3 & e_4 & e_5 \\ d_1 & d_2 & d_3 & d_4 & d_5 \end{pmatrix}$$

que caracteriza o isomorfismo entre os grafos G e H na figura.

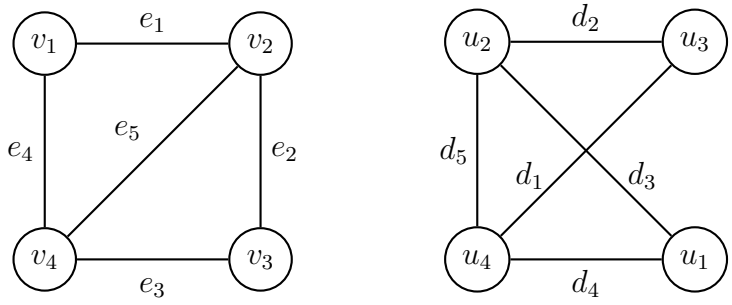


Figura 2.3: Ilustração de dois grafos isomorfos. À esquerda G e à direita H.

Utilizamos o conceito de isomorfismo para caracterizar grafos que embora possuam o rotulamento de seus vértices e arestas diferentes, possuem a mesma estrutura intrínseca. Estamos interessados justamente em tais estruturas para melhor compreensão dos resultados dessa tese. Formalmente definimos um grafo *não-rotulado* como um representante de uma

classe de equivalência de grafos isomorfos. Os grafos descritos na próxima seção serão exemplificados utilizando grafos não-rotulados.

2.1.2 Famílias Especiais de Grafos

Utilizaremos esta subseção para expôr algumas famílias importantes de teoria dos grafos. Um grafo *completo* com n vertices, denotado por K_n, é um grafo tal que quaisquer dois vértices distintos de K_n sejam vizinhos. Um conjunto de vértices que induz um grafo completo é chamado de *clique*. Para um grafo G qualquer denotamos por $\omega(G)$ o número de vértices da maior clique em G. Um grafo G é *vazio* se quaisquer dois vértices distintos de G não sejam vizinhos entre si. Note que um grafo vazio de tamanho n pode ser visto como o complementar de um K_n, portanto denotamos um grafo vazio de ordem n como $\overline{K}_n$. Veja a Figura 2.4 para um exemplo de ambos.

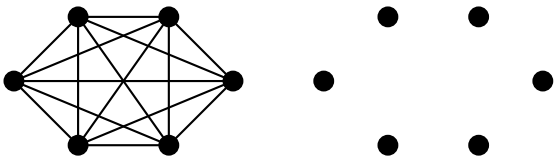


Figura 2.4: À esquerda um K_6 e à direita o seu complementar, $\overline{K}_6$.

Um grafo é *bipartido* se o seu conjunto de vértices pode ser particionado em dois subconjuntos X e Y de tal forma que qualquer aresta possua uma extremidade em X e a outra em Y. Chamaremos a partição (X,Y) de *bipartição* do grafo, e X e Y suas partes. Se temos um grafo bipartido G com bipartição (X, Y) onde X possui m vértices, Y possui n vértices e todo vértice de X é vizinho de todo vértice de Y, então temos um grafo *bipartido completo* e o denotamos por $K_{m,n}$. Podemos visualizar os dois grafos bipartidos na Figura 2.5.

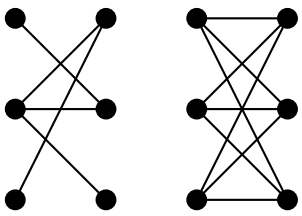


Figura 2.5: À esquerda um grafo bipartido G e à direita um grafo bipartido completo $K_{3,3}$.

Um grafo é um *caminho* se seus vértices podem ser ordenados em uma sequência linear de tal forma que dois vértices sejam vizinhos se eles forem consecutivos na sequência, e são não

vizinhos caso contrário. Denotamos um caminho com n vértices por P_n ou por n-caminho. Um *ciclo* é um grafo cujos vértices podem ser ordenados em uma ordem cíclica de tal forma que dois vértices sejam vizinhos se eles forem consecutivos na sequência. Qualquer ciclo tem no mínimo três vértices e um ciclo de tamanho n é denotado por C_n ou n-ciclo. O *comprimento* de um caminho e um ciclo é a quantidade de arestas que cada um possui. Um exemplo de cada pode ser visto na Figura 2.6.

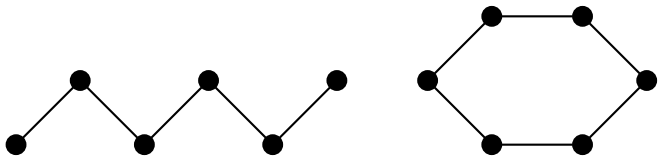


Figura 2.6: À esquerda um caminho P_6 e à direita um ciclo C_6.

Um grafo é *conexo* se para cada partição dos seus vértices em dois conjuntos não vazios X e Y, exista pelo menos uma aresta com uma extremidade em X e a outra em Y. Um grafo que não é conexo é chamado de *desconexo*. Usando a Figura 2.6 como exemplo, se considerarmos somente o P_6 da figura como o grafo analisado, então temos um grafo conexo, porém se estivermos, por exemplo, analisando um grafo que seja a união disjunta de um P_6 e um C_6, então esse grafo é desconexo.

2.1.3 Coloração em Grafos

Uma *k-coloração* em um grafo G é um mapeamento $f : V \to C$, onde $C = \{1, 2, ..., k\}$ é um conjunto de k cores distintas. Uma k-coloração é *própria* se não existem vizinhos com a mesma cor. Nessa dissertação estamos apenas interessados em k-colorações próprias, então diremos que um grafo G é *k-colorível* se existir uma k-coloração própria em G.

O menor inteiro k tal que um grafo G é k-colorível é chamado de *número cromático* e o denotamos por $\chi(G)$. Podemos observar um exemplo de um grafo com $\chi(G) = 3$ na Figura 2.7. Note que não podemos fazer uma coloração própria com menos cores pois existe uma clique de tamanho 3 induzida no grafo, e como todos os vértices da clique são vizinhos, necessitamos de pelo menos 3 cores em G.

Para qualquer grafo G, a desigualdade $\omega(G) \leq \chi(G) \leq \Delta(G) + 1$ é válida pelo fato da maior clique induzida de G necessitar de pelo menos $\omega(G)$ cores para ser propriamente colorida e qualquer vértice de G possuir no máximo $\Delta(G)$ vizinhos, então mesmo se todos

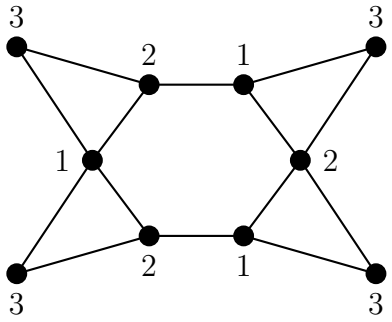


Figura 2.7: Um grafo G com número cromático $\chi(G) = 3$. Como $\omega(G) = 3$, não podemos fazer uma coloração própria com menos cores.

os vizinhos de G possuem cores distintas, basta usar a cor $Delta(G) + 1$ que ainda não foi usada.

2.2 Complexidade Computacional e Lógica Booleana

Chamamos de ***P*** a classe de problemas que podem ser resolvidos em tempo polinomial em uma máquina de Turing determinística. Definimos como *problemas de decisão* os problemas que só possuem resposta SIM ou NÃO. A classe dos problemas de decisão cujas instâncias que possuem resposta "SIM" podem ser verificadas em tempo polinomial é chamada de ***NP***.

Um problema B tal que qualquer problema A em NP possa ser reduzido à B em tempo polinomial é dito ser ***NP-difícil***. Um problema é ***NP-completo*** se pertence à NP e é NP-difícil.

A classe ***PSPSACE*** é composta por todos os problemas de decisão que podem ser resolvidos em **espaço** polinomial em uma máquina de Turing determinística. Um problema B tal que qualquer problema A em PSPACE possa ser reduzido à B em tempo polinomial é dito ser ***PSPACE-difícil***. Um problema é ***PSPACE-completo*** se pertence à PSPACE e é PSPACE-difícil.

A seguinte relação é uma conjectura sobre como os conjuntos de problemas citados se relacionam

$$P \subseteq NP \subseteq PSPACE.$$

suspeita-se também que problemas PSPASCE-completo estejam fora dos conjuntos de problema P e NP, mas ainda é desconhecido que isso seja verdade.

Uma variável que só admite os valores FALSO ou VERDADEIRO é chamada de variável *booleana*. Utilizaremos 0 para representar falso e 1 para verdadeiro. Uma *fórmula*

booleana é uma expressão envolvendo variáveis booleanas e operadores lógicos. Chamaremos as variáveis booleanas nas fórmulas de *literais*. Podemos tomar como exemplo a seguinte fórmula booleana

$$\phi = (x \vee y) \wedge (\overline{x} \vee z)$$

onde $\overline{x}$ representa a negação de x, isto é, se x é 1 então $\overline{x}$ é 0 e vice-versa. Uma fórmula booleana Φ é *satisfazível* se existe uma atribuição de valores nos literais de ϕ de tal forma que o resultado da fórmula seja verdadeiro. Uma fórmula booleana é da *forma normal conjuntiva* se for uma conjunção de uma ou mais cláusulas onde cada cláusula é uma disjunção de literais, como é o caso do exemplo acima.

2.2.1 QSAT

Como exemplo de um problema que pertence a PSPSACE, mostramos o seguinte problema chamado de QSAT (*quantified satisfiability*). Seja Φ uma fórmula booleana na forma normal conjuntiva com literais $x_1, ..., x_n$. A seguinte fórmula é satisfazível?

$$\exists x_1 \; \forall x_2 \; \exists x_3 \; \forall x_4 \; \cdots \; \forall x_{n-1} \; \exists x_n \; \Phi.$$

A expressão acima é lida como "existe um x_1 para todo x_2 existe um x_3 para todo x_4 ... para todo x_{n-1} existe um x_n tal que Φ é satisfazível". QSAT pode ser interpretado como um jogo entre dois jogadores, Alice e Bob que tomam turnos alternados escolhendo sempre o próximo literal da sequência $x_1, ..., x_n$, começando por x_1 e marcando o literal da vez como verdadeiro ou falso. Se após n turnos a expressão Φ resulta em verdadeiro, Alice ganha e Bob ganha caso contrário. Tome como exemplo a seguinte instância de QSAT

$$\exists x_1 \; \forall x_2 \; \exists x_3 \; (x_1 \vee x_2) \wedge (x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3).$$

Alice ganha fazendo x_1 ser verdadeiro, então Bob faz x_2 ser verdadeiro ou falso e Alice faz x_3 ser o mesmo valor de x_2. Como as cláusulas $(x_2 \vee \overline{x_3})$ e $(\overline{x_1} \vee \overline{x_2} \vee x_3)$ ambas possuem x_2, x_3 e suas respectivas negações e Alice garante que a cláusula $(x_1 \vee x_2)$ seja verdadeira ao fazer x_1 ser verdadeiro, ela garante que a fórmula é satisfazível não importando como Bob jogue.

O problema pertence a PSPSACE pois basta testar recursivamente todas as possibilidades. Só é necessário 1 bit de informação para cada subproblema e a quantidade de espaço necessária para resolver o jogo é polinomial pois cada turno possui apenas duas jogadas possíveis: verdadeiro ou falso.

2.2.2 POS CNF

Proposto por Schaefer em (SCHAEFER, 1978) e provado ser um problema PSPSACE-completo, o jogo conhecido como POS CNF (*Positive Conjunctive Normal Form*) tem como entrada uma fórmula booleana ϕ na forma conjuntiva normal sem a ocorrência de negações. Dois jogadores, Alice e Bob, tomam turnos alternados escolhendo um literal presente em ϕ com Alice jogando primeiro. Alice deve sempre atribuir o valor 1 para o literal escolhido e o Bob deve sempre escolher o valor 0. O jogo termina quando todas os literais de ϕ são escolhidos. Alice ganha se ao final do jogo a fórmula ϕ for verdadeira e Bob ganha caso contrário.

Tomando como exemplo a seguinte fórmula

$$\phi = x_1 \wedge (x_2 \vee x_3) \wedge (x_2 \vee x_4)$$

podemos notar que cláusulas com apenas um literal são prioridade para Alice, caso contrário Bob joga em tal literal e ganha o jogo. Para o exemplo, Alice deve jogar primeiro em x_1 e em seguida Bob pode jogar em x_2. Alice então perde o jogo pois ela só pode jogar em um literal de $\{x_3, x_4\}$ e Bob joga no outro, tornando uma das cláusulas falsa.

3 JOGOS DE COLORAÇÃO

São apresentados neste capítulo o jogo de coloração, suas regras e como jogá-lo através de um exemplo simples. Também neste capítulo é apresentado a variação do jogo conhecida como *jogo de coloração gulosa*, mostrando também suas regras e alguns exemplos. Por fim, uma revisão da literatura é feita para ambos os jogos, mostrando os principais resultados sobre ambos os jogos e algumas questões em aberto.

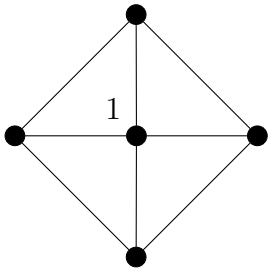
3.1 Jogo de Coloração

Proposto independentemente por Brams em (GARDNER, 1981) como "jogo de coloração em mapas" e por Bodlaender em (BODLAENDER, 1991) como "jogo de construção de coloração", o jogo atualmente é referenciado apenas como *jogo de coloração em grafos* como pode ser visto, por exemplo em (CHARPENTIER; SOPENA, 2013) e (SIA, 2009). Para esta dissertação, nos referimos ao seguinte jogo, que possui as mesmas regras do jogo de construção de coloração de Bodlaender, como "jogo de coloração" podendo ser referenciado também como "versão clássica do jogo de coloração".

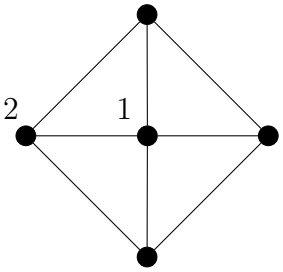
Dado um grafo simples G e um conjunto de cores C com k cores distintas, dois jogadores, Alice e Bob, tomam turnos alternados sempre colorindo um vértice de G por vez. As cores utilizadas devem pertencer ao conjunto C e a coloração deve ser própria, isto é, nunca dois vértices vizinhos possuem a mesma cor. Se após n jogadas (onde n é a quantidade de vértices de G) o grafo for completamente colorido, então Alice é declarada a vencedora. Bob é declarado o vencedor se em algum momento do jogo existir um vértice que não possa mais ser colorido (por exemplo se existir um vértice não colorido com k vizinhos coloridos com cores distintas).

Um exemplo simples de jogo pode ser observado na Figura 3.1 onde temos o grafo G e 3 cores disponíveis. Pela descrição do jogo, Alice faz a primeira jogada e em sua estratégia ela consegue garantir no terceiro turno, após sua jogada, que os vértices restantes podem ser coloridos com a cor 3, então Alice vence o jogo.

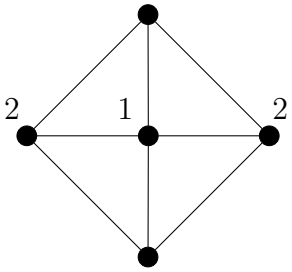
O jogo de coloração pode ter um resultado diferente em um mesmo grafo dependendo de quem faz a primeira jogada. Se considerarmos o mesmo grafo da Figura 3.1 e permitirmos que Bob jogue primeiro, teríamos o possível jogo descrito pela Figura 3.2. Como os vértices restantes possuem vizinhos com as cores $\{1, 2, 3\}$, então Bob ganha o jogo.



(a) Primeiro turno, jogada de Alice.

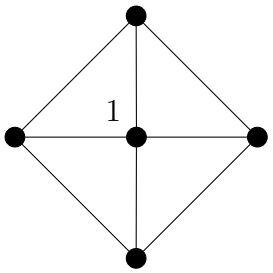


(b) Segundo turno, jogada de Bob.

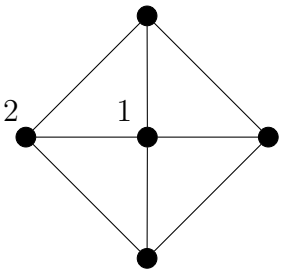


(c) Terceiro turno, jogada de Alice.

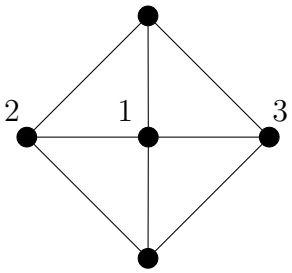
Figura 3.1: Jogo de coloração em G usando 3 cores. Cada figura representa um turno do jogo. As duas jogadas restantes são triviais.



(a) Primeiro turno, jogada de Bob.



(b) Segundo turno, jogada de Alice.



(c) Terceiro turno, jogada de Bob.

Figura 3.2: Jogo de coloração em G usando 3 cores em que Bob faz a primeira jogada.

Para ambas versões em que Alice joga primeiro e em que Bob joga primeiro, iremos definir o seguinte parâmetro.

Definição 3.1. *Seja G um grafo simples. Denotamos por $\chi_g^{(A)}(G)$ a menor quantidade de cores necessárias para Alice garantir uma vitória no jogo de coloração em G quando ela faz a primeira jogada. Analogamente, por $\chi_g^{(B)}(G)$ denotamos a menor quantidade de cores necessárias para que Bob garanta uma vitória quando ele faz a primeira jogada. Os parâmetros são chamados de "número cromático lúdico" de Alice e Bob, respectivamente.*

Um exemplo de grafo onde $\chi_g^{(A)}(G)$ e $\chi_g^{(B)}(G)$ são distintos pode ser visto nas Figuras 3.1 e 3.2, onde temos $\chi_g^{(A)}(G) = 3$ e $\chi_g^{(B)}(G) = 4$.

O número cromático lúdico de Alice foi proposto inicialmente por Bodlaender em (BODLAENDER, 1991) onde era considerado apenas a versão do jogo em que Alice faz a primeira jogada. Quando não estiver explícito qual jogador faz a primeira jogada, assume-se que é Alice o primeiro jogador a jogar e para tal caso denotamos a menor quantidade de cores que Alice necessita para garantir que ela vença o jogo apenas por $\chi_g(G)$, dado um

grafo simples G.

Como a menor quantidade de cores distintas tal que um grafo G possa ser propriamente colorido é $\chi(G)$, Alice não consegue usar menos que $\chi(G)$ cores no jogo de coloração clássico e como qualquer vértice de G não possui mais que $\Delta(G)$ vizinhos, a relação $\chi(G) \leq \chi_g(G) \leq \Delta(G) + 1$ é válida para qualquer grafo G. Os limites superiores e inferiores, entretanto, podem ser bem distantes como é o caso do grafo G na Figura 3.3 onde temos $\chi(G) = 2$ (basta colorir o vértice central com 1 e os restantes com 2) e $\Delta(G) + 1 = 25$.

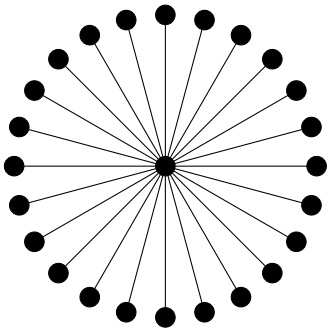


Figura 3.3: Um grafo G com número cromático pequeno e grau máximo alto.

3.2 Jogo de Coloração Gulosa

A variação do jogo coloração conhecida como *jogo de coloração gulosa* foi proposta por Havet em (HAVET; ZHU, 2013). As regras do jogo são semelhantes à versão clássica do jogo.

Seja G um grafo simples. Dois jogadores, Alice e Bob, em turnos alternados devem escolher e colorir um vértice de G ainda não colorido, onde uma cor é um número inteiro e a menor cor é 1, e a condição é que em cada passo a menor cor disponível ainda não usada pela vizinhança do vértice escolhido seja usada para o colorir. O objetivo de Alice é minimizar a quantidade de cores utilizadas em G enquanto o objetivo de Bob é maximizar a quantidade de cores.

Definição 3.2. *O número de Grundy lúdico de Alice $\Gamma_g^{(A)}(G)$ é a menor quantidade de cores distintas usadas ao fim de um jogo na versão em que Alice faz a primeira jogada. Analogamente, o número de Grundy lúdico de Bob $\Gamma_g^{(B)}$ é a menor quantidade de cores usadas ao fim de um jogo na versão em que Bob faz a primeira jogada.*

Quando não especificado quem faz a primeira jogada, assume-se que seja Alice e nesse caso o parâmetro $\Gamma_g(G)$ é usado no lugar do parâmetro $\Gamma_g^{(A)}(G)$.

O parâmetro $\Gamma_g(G)$ foi inspirado no *número de Grundy* denotado por $\Gamma(G)$ e introduzido por Christen & Selkow em (CHRISTEN; SELKOW, 1979). Uma k-coloração de Grundy em um grafo G é uma k-coloração tal que cada vértice colorido com uma cor i possua pelo menos uma vizinho colorido com a cor j para cada $j < i$. Denotamos por $\Gamma(G)$ o maior k tal que G possua uma k-coloração de Grundy.

Mesmo se ambos os jogadores "cooperem" no jogo de coloração gulosa com o objetivo de obter a maior quantidade k de cores possíveis ao final do jogo, ainda assim esse valor k não passa de $\Gamma(G)$. Por outro lado, assim como no jogo de coloração clássica, não importa como os jogadores façam sua coloração, a quantidade mínima de cores usadas ao fim do jogo não é menor do que $\chi(G)$. Podemos concluir que a seguinte desigualdade $\chi(G) \leq \Gamma_g(G) \leq \Gamma(G)$ é válida para qualquer grafo simples G.

Note que da versão clássica para a versão gulosa do jogo de coloração temos duas modificações: não possuímos um conjunto fixo de cores C e a forma de colorir deve ser gulosa. Não é claro se as diferenças entre as versões clássica e a gulosa do jogo de coloração impliquem que o jogo seja mais "fácil" para Alice no sentido de que ela consegue garantir que menos cores são usadas ao final do jogo de coloração gulosa. De fato, Bob é restrito na hora de colorir um vértice, mas a mesma restrição é aplicada à Alice, o que dificulta a validação de tal desigualdade.

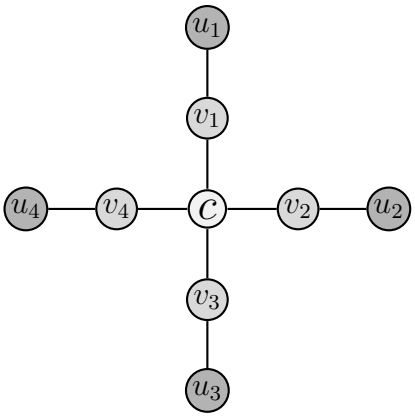


Figura 3.4: Grafo utilizado para mostrar um exemplo do jogo de coloração gulosa.

Para melhor entender o jogo de coloração gulosa iremos tomar como referência o grafo da Figura 3.4. Primeiramente iremos mostrar a versão do jogo em que Alice joga primeiro.

Antes de iniciar o jogo note que o grafo pode ser propriamente colorido com duas cores, basta colorir $v_1, \ldots, v_4$ com a cor 1 e o restante com a cor 2.

Iremos detalhar a estratégia vencedora de Alice em que ela garante no máximo duas cores em G. No início ela irá colorir c com 1. Note agora que todos vértices de $\{v_1, \ldots, v_4\}$ devem ser coloridos com a cor 2 caso sejam eventualmente coloridos pois todos os vértice que não pertencem à $\{v_1, \ldots, v_4, c\}$ precisam ser coloridos com a cor 1 até o final do jogo. Então Alice consegue colorir G usando apenas duas cores e como $2 = \chi(G) \leq \Gamma_g(G)^{(A)} = \Gamma_g(G)$ concluímos que $\Gamma_g(G) = 2$.

Considere agora a versão em que Bob joga primeiro. Se ele jogar em c, teremos o mesmo caso descrito acima, então ele irá escolher um vértice diferente no começo. Primeiramente ele irá escolher um vértice de $\{v_1, \ldots, v_4\}$, digamos v_1 com a cor 1. Independente de onde Alice jogue em seguida, Bob pode colorir um vértice u_i de $\{u_2, u_3, u_4\}$ com a cor 1 e o caminho de u_i até v_1 é um P_4 induzido onde as extremidades possuem a mesma cor, então os vértices centrais desse P_4 devem ser coloridos com cores distintas e diferentes de 1, então Bob consegue garantir pelo menos 3 cores. Um dos possíveis casos pode ser visto na Figura 3.5.

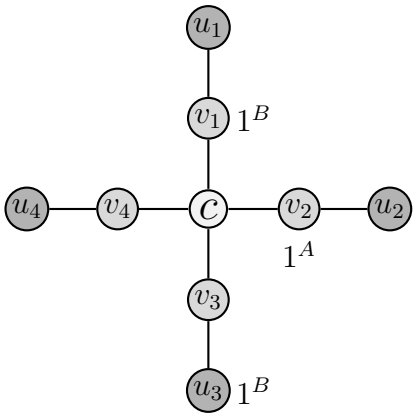


Figura 3.5: Ilustração de uma sequência de jogadas em que Bob garante 3 cores no grafo descrito. O símbolo 1^B indica que o vértice foi colorido por Bob com a cor 1 e 1^A indica que Alice coloriu o vértice com a cor 1. Os vértices C e v_3 serão eventualmente coloridos com cores distintas entre si e ambas diferentes de 1.

Como Bob possui uma sequência de jogadas que garante 3 cores distintas ao final do jogo, independente de como Alice jogue, então $\Gamma_g(G) \leq 3$.

Assim como na versão clássica do jogo, dependendo de quem faça a primeira jogada o número de cores totais utilizadas para colorir um grafo qualquer pode mudar.

3.3 Resultados da Literatura e Perguntas em Aberto

3.3.1 Jogo de coloração clássico

Em (BARTNICKI *et al.,* 2007), Bartnicki, Tomasz & Grytczuk recapitulam os principais resultados e perguntas em aberto sobre o jogo de coloração até o ano de 2007, em especial para grafos planares.

Uma das principais motivações para a concepção do jogo de coloração de Brams em (GARDNER, 1981) foi o seguinte teorema, conhecido como "Teorema das 4 cores", provado inicialmente por Kenneth Appel & Wolfgang Haken em 1976.

Teorema 3.1. *Para qualquer grafo planar P. Temos que $\chi(G) \leq 4$.*

A prova de Kenneth Appel & Wolfgang Haken para Teorema 3.1 foi feita com o uso de máquinas, o que não convenceu parte da comunidade de teoria dos grafos na época. A abordagem de Brams para o problema consiste em tratar a tarefa de descobrir se um grafo planar pode ser colorido ou não com 4 cores como um jogo. O objetivo inicial da abordagem é determinar se parâmetros fornecidos pelo jogo de coloração que possuam algum tipo de relação com parâmetros de coloração de grafos. O objetivo final é encontrar uma prova para o Teorema 3.1 utilizando o conhecimento proveniente da pesquisa de jogos de coloração, se possível sem o uso de máquinas.

Apesar de não obter sucesso em sua abordagem, o trabalho de Brams gerou um novo problema para a comunidade de grafos: Qual a menor quantidade de cores tal que Alice possua uma estratégia vencedora para o jogo de coloração em grafos planares?

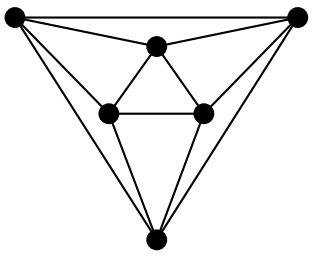


Figura 3.6: Um grafo planar G onde cada vértice corresponde à uma face de um cubo e uma aresta liga dois vértices se as faces correspondentes aos vértices sejam adjacentes no cubo.

Robert High encontrou um exemplo simples em que 5 cores eram necessárias para Alice ganhar o jogo, um grafo planar onde cada vértice corresponde à face de um cubo e dois vértices são adjacentes se a faces correspondentes no cubo sejam vizinhas. O grafo pode

ser visualizado na Figura 3.6. Considerando que cada vértice corresponde a uma face de um cubo, a estratégia de Bob consiste em sempre jogar na face oposta que Alice jogou por último, sempre que possível utilizando uma cor ainda não usada no jogo e, após 4 turnos, os dois vértices restantes possuem 4 vizinhos coloridos com cores distintas, então a quinta e última cor deve ser usada.

Lloyd Shapley encontrou um exemplo de grafo planar onde 6 cores eram necessárias, o grafo da Figura 3.7, que é construído de forma similar ao grafo planar da Figura 3.6, a única diferença é a de que a construção é feita tomando como referência um dodecaedro no lugar de um cubo. A forma como Shapley mostra que 6 cores são necessárias para Alice utiliza uma argumenta similar ao de Robert High no cubo, a única diferença é a de que Bob deve colorir a face oposta que Alice jogou por último utilizando a mesma cor.

Robert High também encontrou exemplos de grafos planares que Alice necessita de 7 para ganhar o jogo de coloração. Um possível exemplo em que 8 cores são necessárias pode ter sido descoberto pelos pesquisadores Tomasz Bartnicki, Jarosław Grytczuk, H. A. Kierstead, e Xuding Zhu porém ainda não pôde ser validado pela grande quantidade de jogadas possíveis que devem ser analisadas.

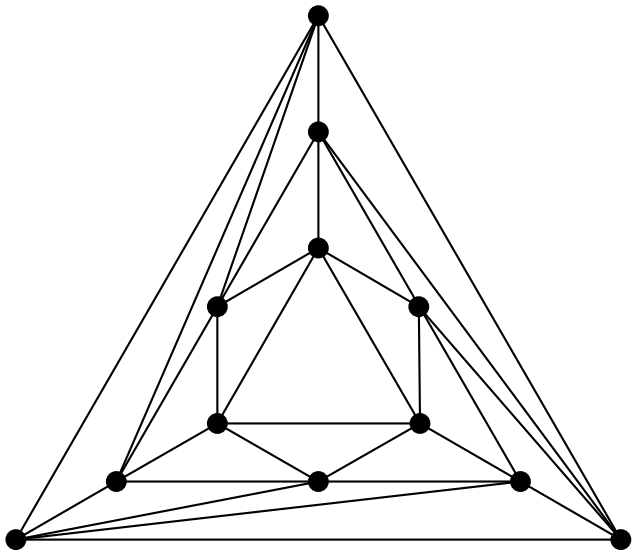


Figura 3.7: Um grafo planar G onde cada vértice corresponde à uma face de um dodecaedro e uma aresta liga dois vértices se as faces correspondentes aos vértices sejam adjacentes no dodecaedro.

Ainda sobre grafos planares, Xuding Zhu mostra em (ZHU, 2008) através de um estudo do chamado "jogo de marcação" que $\chi_g(G) \leq 17$ para grafos planares, que em conjunto com

os exemplos comprovados de grafos planares de Robert High em que $\chi_g(G) \geq 7$ mostram que $7 \leq \chi_g(G) \leq 17$. O jogo de marcação utilizado por Zhu na prova do limite superior de $\chi_g(G)$ para grafos planares é descrito como dois jogadores, Alice e Bob, tomando turnos alternados "marcando" um vértice ainda não marcado em um grafo G que inicialmente não possui vértice marcados. Durante toda a duração do jogo é registrado a maior quantidade de vizinhos marcados que qualquer vértice não marcado de G possui até o momento e chamaremos esse valor de z. Se Alice consegue garantir que z não ultrapasse um valor fixo k, então ela vence o jogo. Define-se então como $col_g(G)$ o menor valor $k+1$ tal que Alice garanta que z não ultrapasse k.

O jogo de marcação e o parâmetro $col_g(G)$ foram criados por Erdös & Hajnal e em (ERDOS; HAJNAL, 1966) para auxiliar o estudo do número cromático em grafos e eventualmente no estudo do jogo de coloração. Dado um grafo G e uma estratégia vitoriosa para Alice no jogo de marcação, Alice pode se utilizar dessa estratégia para garantir sua vitória no jogo de coloração quando ele é jogado com $col_g(G)$ cores. Note que, toda vez que um jogador marca um vértice no jogo de marcação esse mesmo vértice tem no máximo $col_g(G) - 1$ vizinhos marcados. Dessa forma, se Alice pode adaptar a estratégia do jogo de marcação para o jogo de coloração clássico, tendo a garantia de que qualquer vértice escolhido por ela ou por Bob tem no máximo $col_g(G) - 1$ vizinhos coloridos. Assim, sempre que ela ou Bob escolhem um vértice para colorir, existe pelo menos uma cor disponível. Com esse argumento é possível mostrar que $\chi_g(G) \leq col_g(G)$ para todo grafo G.

Vários dos limites e valores para $\chi_g(G)$ na literatura são obtidos por intermédio do jogo de marcação. Em (ZHU, 1999), (ZHU, 2000) e (SIDOROWICZ, 2007) é possível encontrar esses limites para grafos planares, k-árvores parciais e cacti respectivamente.

Além de grafos planares, o parâmetro $\chi_g(G)$ também foi estudado para uma variedade de família de grafos. Faigle et al. mostraram em (FAIGLE *et al.,* 1993) que $\chi_g(G) \leq 4$ quando G é uma floresta e Sidorowicz em (SIDOROWICZ, 2007) mostrou que $\chi_g(G) \leq 5$ para cactus. Kierstead & Trotter provaram em (KIERSTEAD; TROTTER, 1994) que $\chi_g(G) \leq 7$ para grafos periplanares. Dinski & Zhu provaram em (DINSKI; ZHU, 1999) que $\chi_g(G) \leq k(k+1)$ para todo grafo G com número cromático acíclico k. Em (ZHU, 2000), Zhu provou que $\chi_g(G) \leq 3k+2$ para k-árvores parciais. Em (BOHMAN *et al.,* 2008), Bohman, Frieze & Sudakov investigaram o comportamento assintótico de $\chi_g(G_{n,p})$ para o grafo aleatório $G_{n,p}$.

Desde sua concepção até o atual ano desta dissertação, apesar de ter sido vastamente

estudado nos últimos anos, ainda não se sabe a complexidade do jogo de coloração. Em (DUNN *et al.,* 2015) no ano de 2015 Dunn, Larsen, Lindke, Retter & Toci investigam a complexidade do jogo de coloração para árvores e florestas, obtendo resultados parciais. Segundo suas próprias palavras "mais de duas décadas depois e essa questão continua aberta".

Um outro problema em aberto foi a seguinte pergunta proposta por Zhu em (ZHU, 1999).

- (Questão 1) Suponha que $\chi_g(G) = k$. É verdade que para qualquer $k' > k$, se o conjunto de cores disponíveis C possui cardinalidade k', então Alice possui uma estratégia vencedora para o jogo de coloração em G?

Por um lado é intuitivo pensar que seja vantajoso para Alice poder jogar com mais cores, mas por outro lado Bob também possui mais opções de cores, o que dificulta a análise do jogo.

3.3.2 Jogo de coloração gulosa

Em 2013 quando propostos por Havet & Zhu em (HAVET; ZHU, 2013), o jogo de coloração gulosa e o número de Grundy lúdico $\Gamma_g(G)$ são estudados para algumas classes de grafos. Dentre os resultados, Havet mostra que $\Gamma_g(G) \leq 3$ para florestas, $\Gamma_g(G) \leq 7$ para 2-árvores parciais e que $\Gamma_g(G) = \chi(G)$ quando G é um cografo. Também são propostas as seguintes perguntas a respeito do número de Grundy lúdico:

- Problema 5 de (HAVET; ZHU, 2013): $\chi_g(G)$ pode ser limitado por uma função de $\Gamma_g(G)$?
- Problema 6 de (HAVET; ZHU, 2013): É verdade que $\Gamma_g(G) \leq \chi_g(G)$ para qualquer grafo G?

Em 2015, Krawczyk & Walczak respondem em (KRAWCZY; WALCZAK, 2015) o Problema 5 de (HAVET; ZHU, 2013), mostrando que $\chi_g(G)$ não é limitado por uma função de $\Gamma_g(G)$. De nosso conhecimento, o Problema 6 de (KRAWCZY; WALCZAK, 2015) continua em aberto.

Assim como na versão clássica do jogo de coloração, a análise de complexidade do jogo de coloração gulosa também é uma questão em aberto.

4 ANÁLISE DE COMPLEXIDADE

Neste capítulo mostramos uma análise de complexidade para ambas versões clássica e gulosa do jogo de coloração. Ambas versões possuem a mesma prova de que são PSPACE, então iremos propor o seguinte lema e sua prova para ser usado posteriormente na prova de que ambos os jogos são PSPACE-Completos.

Lema 4.1. *Decidir se Alice possui uma estratégia vencedora usando k cores, para a versão clássica ou gulosa do jogo de coloração, é um problema que pertence à PSPACE.*

Demonstração. Seja $G = (V, E)$ um grafo simples com $|V| = n$ onde o jogo de coloração clássico ou o jogo de coloração gulosa será jogado. Temos k cores que podem ser usadas. A partir do grafo G criamos um grafo H que representa uma árvore de decisões para o jogo de coloração em G.

No começo H possui apenas um vértice r que será o nó raiz. Criamos então $n \cdot k$ vértices que se ligam à r onde cada um representa o vértice que foi selecionado no primeiro turno do jogo em G e com qual cor o vértice foi colorido. Chamaremos esse primeiro conjunto de vértices, menos r, de "primeiro nível de profundidade de H".

Para cada vértice v do primeiro nível de profundidade de H, criamos $(n-1) \cdot k$ vértices que são conectados à v que representam o vértice escolhido no segundo turno do jogo e a cor utilizada para colorir o vértice. Para cada profundidade i seguinte criamos sempre $(n-i) \cdot k$ vértices para cada vértice u da i-ésima profundidade criada, ligando os $(n-i) \cdot k$ vértices à u. Paramos o procedimento quando obtivermos n níveis de profundidades. A Figura 4.1 possui um exemplo da criação do grafo H para um grafo G com dois vértices e duas cores disponíveis.

Note que estamos enumerando todas as colorações (não necessariamente próprias) do grafo G usando k cores distintas. Para um grafo G qualquer e k cores distintas, a lista de todos os jogos possíveis, isto é, todas as combinações de jogadas alternadas possíveis entre Alice e Bob em G, pode ser enumerado fazendo uma busca em largura em H, sempre verificando se o nó visitado atualmente traduz em uma coloração própria no grafo G e, se não for o caso, diremos que a jogada atual é "não permitida". Se a coloração atual é própria, então dizemos que a jogada atual é "permitida".

Note que o espaço total necessário para terminar de executar a busca em largura em H é apenas o espaço necessário para encontrar todas as colorações próprias em G usando k cores

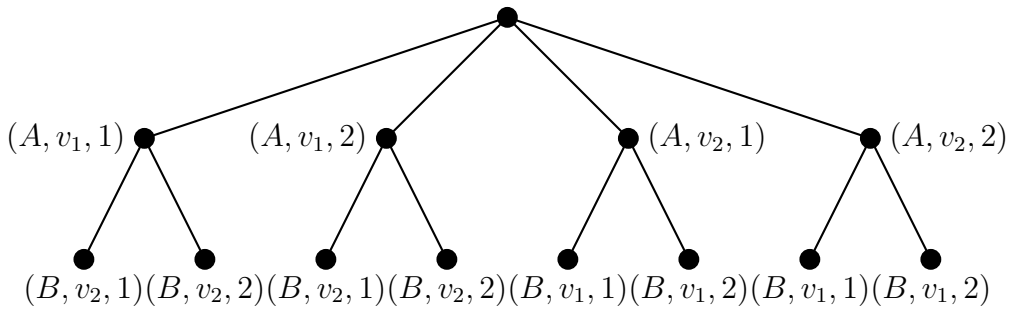


Figura 4.1: Grafo H que representa uma árvore de decisões para o jogo de coloração em um K_2 (dois vértices v_1 e v_2 que são vizinhos) e duas cores disponíveis. A notação na figura representa, em ordem, o jogador (onde A é Alice e B é Bob), o vértice colorido e a cor usada.

e o que for necessário para determinar se existe uma sequência de jogadas para Alice tal que, independente de como Bob jogue, ela consiga colorir G propriamente usando k cores. Para ambos só precisamos usar espaço linear na quantidade de vértices do grafo G, visto que para enumerar qualquer jogada permitida do jogo de coloração clássico em G basta escolher um vértice v de G qualquer ainda não colorido e atribuir uma das k cores disponíveis no jogo. Analogamente, para o jogo de coloração gulosa basta escolher um vértice v ainda não colorido e atribuir a menor cor disponível para v que não aparece em sua vizinhança.

□

4.1 Complexidade da versão clássica

Relembre que por "versão clássica" nos referimos ao jogo de coloração de Bodlaender. Para nossa redução iremos primeiramente definir dois problemas decisão para o jogo de coloração: dados um grafo G e um inteiro k,

- (Problema 1) $\chi_g(G) \leq k$?
- (Problema 2) Alice possui uma estratégia vencedora com k cores?

E a seguinte questão proposta por Zhu em (ZHU, 1999) (como na página 20 desta dissertação).

- (Questão 1) Suponha que $\chi_g(G) = k$. É verdade que para qualquer $k' > k$, se o conjunto de cores disponíveis C possui cardinalidade k', então Alice possui uma estratégia vencedora para o jogo de coloração em G?

Os Problemas 1 e 2 são equivalentes se e somente se a Questão 1 for verdade. Iremos mostrar que o seguinte Problema 3 é PSPACE-Completo dado um grafo G e seu número cromático $\chi(G)$,

- (Problema 3) $\chi_g(G) = \chi(G)$?

Note que os Problemas 1 e 2 são generalizações do Problema 3, já que ambos são equivalentes à ele para $k = \chi(G)$, isso porque $\chi_g(G) \leq k = \chi(G)$ se e somente se $\chi_g(G) = \chi(G)$, o que é verdade se e somente se Alice possui uma estratégia vencedora com $k = \chi(G)$ cores. Portanto, se o Problema 3 é PSPACE-Difícil, então os Problemas 1 e 2 também são PSPACE-Difíceis.

Nossa redução parte de uma versão modificada do problema POS CNF (introduzido no final do Capítulo 1) em que temos uma adição às regras: Bob pode passar sua vez. As reduções que propomos para ambas versões clássica e gulosa do jogo de coloração partem do problema de determinar se Alice possui ou não uma estratégia vencedora no POS CNF na versão em que Bob pode passar a sua vez, e para isso temos que primeiramente mostrar que essa versão modificada não muda a forma que Alice joga o jogo de POS CNF.

Sabemos que o problema POS CNF, na versão em que ambos os jogadores não podem passar sua vez, é um problema PSPACE-Completo como pode ser visto em (SCHAEFER, 1978). Dito isso, temos o seguinte lema.

Lema 4.2. *Alice possui um estratégia vencedora no jogo POS CNF se e só se ela posssui estratégia vencedora no jogo em que Bob pode passar a vez.*

Demonstração. Seja F uma formula booleana na forma normal conjuntiva em que o jogo de POS CNF será jogado. Suponha que Alice possui uma estratégia vencedora em F quando Bob não pode passar a sua vez, então em algum momento do jogo todas as cláusulas de F possuem pelo menos um literal com o valor Verdade. Se Bob pode passar sua vez, então Alice ainda possui sua estratégia vencedora pois se ela garante que cada cláusula tenha pelo menos um literal com o valor Verdade, então quando Bob passa a sua vez, ela ainda pode jogar nos mesmos literais que ela joga em sua jogada vencedora na versão em que Bob não passa a vez e ao fim do jogo teremos mais literais com o valor Verdade e pelo menos um literal em cada cláusula com o valor Verdade.

Suponha agora que Bob possui uma estratégia vencedora no POS CNF, então se ele pode passar a sua vez, basta que ele utilize a mesma estratégia quando ele não poderia passar a vez. □

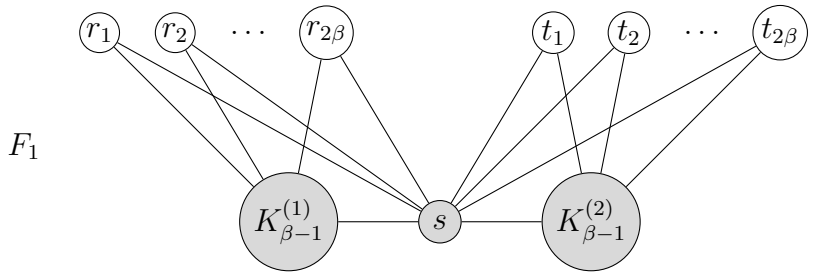


Figura 4.2: Grafo F_1. $K^{(1)}_{\beta-1}$ e $K^{(2)}_{\beta-1}$ são ambas cliques de tamanho $\beta - 1$. Cada vértice de r_1 à $r_{2\beta}$ é vizinho de cada vértice de $K^{(1)}_{\beta-1}$, cada vértice de t_1 à $t_{2\beta}$ é vizinho de cada vértice de $K^{(2)}_{\beta-1}$ e s é vizinho de todos os vértices de F_1.

Para as reduções seguintes iremos utilizar na redução o jogo de POS CNF em que Bob pode passar a sua vez.

Antes de começarmos a redução iremos introduzir o grafo F_1 da Figura 4.2 que possui a propriedade interessante de que para uma quantidade específica de cores disponíveis, quem faz a primeira jogada é quem ganha o jogo.

O grafo F_1 é composto pelos conjuntos de vértices $I^{(1)} = \{r_1, ..., r_{2\beta}\}$, $I^{(2)} = \{t_1, ..., t_{2\beta}\}$, $K^{(1)}_{\beta-1}$, $K^{(2)}_{\beta-1}$ e um vértice especial s, onde β é um número inteiro. $I^{(1)}$ e $I^{(2)}$ são conjuntos independentes de tamanho 2β, $K^{(1)}_{\beta-1}$ e $K^{(2)}_{\beta-1}$ são cliques de tamanho $\beta - 1$ e o vértice s é universal, isto é, s é vizinho de todos os vértices de F_1. Todo vértice de $I^{(1)}$ é vizinho de todo vértice de $K^{(1)}_{\beta-1}$ e todo vértice de $I^{(2)}$ é vizinho de todo vértice de $K^{(2)}_{\beta-1}$.

Lema 4.3. *Alice possui uma estratégia vencedora para o grafo F_1 da Figura 4.2 no jogo de coloração com $2\beta - 1$ cores se e somente se ela é a primeira a jogar. Nesse caso, ela deve colorir o vértice s primeiro. Além disso, uma estratégia vencedora ainda existe mesmo se Bob puder passar a vez em seu turno.*

Demonstração. Sem perda de generalidade, suponha que Alice em sua primeira jogada irá colorir algum vértice em $K^{(2)}_{\beta-1} \cup I^{(2)}$. Então, Bob possui uma estratégia vencedora que consiste em sempre em seu turno colorir algum vértice de $I^{(1)}$ sempre com uma cor nova. Ao fim de sua β-ésima jogada, Bob terá usado β cores distintas em $I^{(1)}$. Como $K^{(1)}_{\beta-1} \cup \{s\}$ é uma clique de tamanho β onde todo vértice é vizinho de $I^{(2)}$, então não é possivel colorir F_1 com $2\beta - 1$ cores.

Iremos agora mostrar que Alice possui uma estratégia vencedora em F_1 usando $2\beta - 1$ cores. Em seu primeiro turno, Alice irá colorir o vértice s com a cor 1. Para cada jogada subsequente, Alice irá colorir um vértice usando a menor cor disponível seguindo a seguinte

ordem de prioridade:

1. Se Bob colorir um vértice de $K^{(i)}_{\beta-1} \cup I^{(i)}$ então ela irá colorir um vértice de $K^{(i)}_{\beta-1}$;

2. Se todos os vértices de uma clique $K^{(i)}_{\beta-1}$ estiverem coloridos, ela irá colorir um vértice da outra clique;

3. Se todos os vértices de $K^{(1)}_{\beta-1} \cup K^{(2)}_{\beta-1}$ estiverem coloridos, ela irá colorir qualquer outro vértice de F_1.

Sem perda de generalidade suponha que ambos jogadores jogam apenas em vértices de $K^{(1)}_{\beta-1} \cup I^{(1)} \cup \{s\}$. Considere o turno em que $K^{(1)}_{\beta-1}$ acaba de ser completamente colorido. Note que, nesse turno, Bob jogou no máximo $\beta - 1$ vezes. Então, foram usadas β cores em $K^{(1)}_{\beta-1} \cup \{s\}$ e no máximo $\beta - 1$ cores em $I^{(1)}$. Como cada vértice de $I^{(1)}$ pode compartilhar qualquer cor já usada em $I^{(1)}$, foram usadas no máximo $\beta + \beta - 1$ cores e todos os vértice não coloridos de $I^{(1)}$ podem ser coloridos com cores já usadas em $I^{(1)}$. □

Teorema 4.1. *Dado um grafo G, decidir se $\chi_g(G) = \chi(G)$ é um problema PSPACE-Completo. Consequentemente, se k é um inteiro, decidir se $\chi_g(G) \leq k$ ou decidir se Alice possui uma estratégia vencedora com k cores são problemas PSPACE-Completos.*

Demonstração. Pelo Lema 4.1 o problema pertence à PSPACE, resta mostrar que o problema é PSPACE-difícil. Dada uma fórmula POS CNF com N variáveis $X_1, \ldots, X_N$ e M cláusulas $C_1, \ldots, C_M$, seja p_m (para $m \in \{1, \ldots, M\}$) o tamanho da cláusula C_m e seja q_i (para $i \in \{1, \ldots, N\}$) o número de cláusulas contendo a variável X_i. Seja $p = \max_{m=1,\ldots,M}\{p_m\}$ e $q = \max_{j=1,\ldots,N}\{q_j\}$. Isto é, cada cláusula tem no máximo p variáveis e toda variável aparece no máximo em q cláusulas. Seja $\beta = \max\{p, q, 4N\}$. Iremos construir um grafo G tal que $\chi(G) = 2\beta - 1$ e $\chi_g(G) = \chi(G)$ se e somente se Alice possui uma estratégia vencedora para a fórmula do POS CNF.

Para construir o grafo G comece apenas com o grafo F_1 da Figura 4.2. Adicione um novo vértice y a G. Para cada variável X_i, crie um vértice x_i em G, chamaremos tais vértices de "vértices variáveis". Para cada cláusula $C_m \in \{1, ..., M\}$ do POS CNF, crie uma clique D_m com vértices $\ell_{m,1}, ..., \ell_{m,p_m}$, chamaremos tais cliques por "clique cláusula". Cada clique cláusula D_m possui p_m vértices, que é a quantidade de variáveis presentes na cláusula C_m do POS CNF.

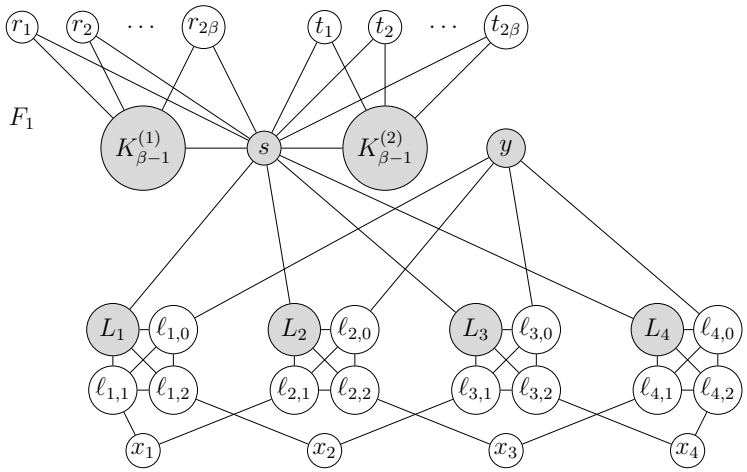


Figura 4.3: Grafo G construído para a fórmula $(X_1 \vee X_2) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_4) \wedge (X_3 \vee X_4)$. Lembre-se que cada vértice $\ell_{m,k}$ representa dois gêmeos-verdadeiros $\ell'_{m,k}$ e $\ell''_{m,k}$; L_1, L_2, L_3, L_4 são cliques com 25 vértices. Bob possui uma estratégia vencedora com 31 cores.

Para cada $i \in \{1, ..., N\}$, crie arestas que conectam x_i à exatamente um vértice da clique cláusula D_m se e somente se a variável X_i do POS CNF aparece na cláusula C_m da fórmula do POS CNF. Crie também o vértice $\ell_{m,0}$ (que não é associada à variáveis) e junte-o por uma aresta à y e todos os vértices da clique cláusula D_m. Para cada vértice $\ell_{m,k}$ ($m = 1, ..., M$ e $k = 0, ..., p_m$), substitua-o por dois vértices gêmeos-verdadeiros, que são vizinhos entre si, $\ell'_{m,k}$ e $\ell''_{m,k}$ que possuem a mesma vizinhança de $\ell_{m,k}$. Além disso, para $m = 1, ..., M$ adicione uma clique L_m de tamanho $2(\beta - p_m) - 3$ e adicione todas as arestas que conectam vértices de $L_m \cup \{\ell_{m,0}\}$ à D_m, formando assim M cliques de tamanho $2\beta - 1$ que chamaremos de "cliques mestre". Por fim, crie todas as arestas sv_i onde s é o vértice s de F_1 e $v_i \in L_1 \cup ... \cup L_M$. Com isso, teremos M cliques mestre de tamanho $2\beta - 1$.

A Figura 4.3 esboça o grafo construído G para a fórmula $(X_1 \vee X_2) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_4) \wedge (X_3 \vee X_4)$. Note que nesse exemplo Bob possui uma estratégia vencedora para o POS CNF: se Alice faz X_1 Verdadeiro, Bob faz X_4 Falso; se Alice faz X_4 Verdadeiro, Bob faz X_1 Falso; se Alice faz X_2 Verdadeiro, Bob faz X_3 falso; se Alice faz X_3 Verdadeiro, Bob faz X_2 Falso. Na redução para este exemplo, temos $N = 4$ variáveis, $M = 4$ cláusulas, $p = 2$, $q = 2$, $\beta = 4N = 16$, as cliques L_1 à L_M possuem $2(\beta - p) - 3 = 25$ vértices cada e cada clique mestre possui 31 vértices.

É fácil verificar que $\chi(G) = 2\beta - 1$. Comece colorindo s com 1, faça o vértice y e todo vértice x_i ($i = 1, \ldots, N$) ter a cor $2\beta - 1$. Os vértices de cada clique de F_1 podem ser coloridos com as cores de 2 à β e cada vértice que pertence à um conjunto indepedente de

F_1 pode ser colorido usando a cor $\beta + 1$. Para cada $m = 1, \ldots, M$, faça os vértices $\ell'_{m,k}$ e $\ell''_{m,k}$ terem as cores $2k + 1$ e $2k + 2$ ($k = 0, \ldots, p_m$). Finalmente, faça os vértices de L_m terem as cores $2p_m + 3, \ldots, 2\beta - 1$. Como cada clique mestre possui $2\beta - 1$ vértices, então $\chi(G) = 2\beta - 1$.

A seguir, mostramos que Alice possui uma estratégia vencedora na versão clássica do jogo de coloração se e somente se ela possui uma estratégia vencedora no POS CNF. Pelo Lema 4.3, em sua primeira jogada, Alice deve colorir o vértice s de F_1, caso contrário, F_1 não pode ser colorido com $2\beta - 1$ cores. Note também que todos os vértices variáveis tem grau no máximo $2q < 2\beta - 1$ então podem sempre ser coloridos com uma cor em $\{1, \ldots, 2\beta - 1\}$. Iremos mostrar que o vértice y será colorido nos três primeiros turnos. Para garantir uma coloração própria usando as cores de $\{1, \ldots, 2\beta - 1\}$, Alice deve necessariamente garantir que todas as cores aparecendo na vizinhança exterior de uma clique mestre também apareça dentro da clique mestre.

Para ilustrar a condição necessária de vitória de Alice, o grafo parcialmente colorido da Figura 4.4 mostra um caso em que não é possível colorir os vértices da clique com as cores que aparecem em vértices da vizinhança exterior da clique. Dentre os vértices da clique, seis deles possuem vizinhos fora da clique com a cor 1 e dois deles possuem vizinhos fora da clique com a cor 2. Como os vértices ainda não coloridos possuem a cor 1 em sua vizinhança, não podemos usar a cor 1, porém ainda não usamos a cor 1 dentro da clique, então para o exemplo da figura necessitamos de nove cores para terminar de colorir propriamente o grafo.

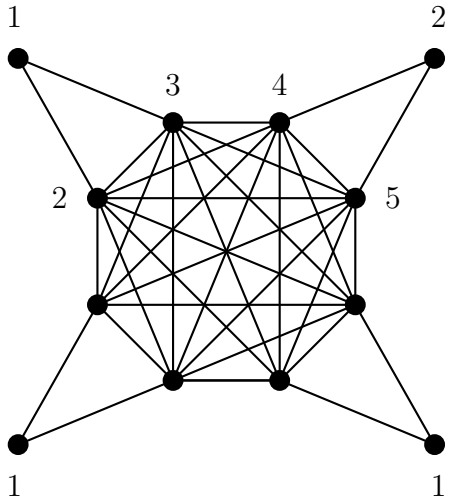


Figura 4.4: Um grafo parcialmente colorido onde os vértices não-coloridos não podem mais ser coloridos com a cor 1.

Para o caso em que Bob possui uma estratégia vencedora no POS CNF, iremos mostrar que a estratégia de Bob é fazer com que todos os vértices na vizinhança exterior de uma clique mestre sejam coloridas com a mesma cor de s, impedindo dessa forma que Alice possa usar essa cor dentro da clique mestre.

Primeiro mostramos que se Bob possui uma estratégia vencedora no jogo de POS CNF, então $\chi_g(G) > 2\beta - 1$. Para a nossa redução, colorir um vértice variável x_i com a cor 1 significa marcar o literal X_i do POS CNF como Falso. Assuma que Bob vence no jogo de POS CNF e Alice começa colorindo o vértice s com a cor 1 (relembre o Lema 4.3). Bob pode usar a seguinte estratégia. Em seu primeiro turno, ele irá colorir y com 1, e então para cada turno subsequente ele segue a seguinte lista de prioridades:

1. se Alice colorir um vértice em $N[x_i]$ (onde x_i é um vértice variável), Bob considera que ela marcou X_i como Verdadeiro no POS CNF e com a cor 1 ele irá colorir o vértice x_j representando o literal X_j escolhido por ele em sua estratégia vencedora no POS CNF;

2. se Alice não colorir um vértice em $\cup_{i=1}^{N} N[x_i]$ (união da vizinhança fechada de todos os vértices variável), então Bob joga como se Alice tivesse passado seu turno no POS CNF. Como Bob possui uma estratégia vencedora no jogo de POS CNF mesmo quando seu oponente pode passar a vez, em algum ponto todos os literais de alguma cláusula estarão marcados como Falsos. Isto significa que todos os vizinhos exteriores de uma clique mestre estão coloridos com a cor 1 (visto que s e y estão coloridos com a cor 1), impossibilitando que essa clique possa ser colorido com a cor 1. Como cada clique mestre possui $2\beta - 1$ vértices e a cor 1 não pode ser usada, necessitamos de 2β cores para terminar de colorir propriamente a clique mestre e $\chi_g(G) > 2\beta - 1$.

Mostramos agora que se Alice possui uma estratégia vencedora no jogo de POS CNF então $\chi_g(G) = 2\beta - 1$. Assuma que Alice ganha no POS CNF e que ela irá colorir o vértice s de F_1 com a cor 1 em sua primeira jogada.

Suponha primeiramente que Bob irá colorir um vértice u diferente de y em seu primeiro turno. Iremos supor que y e qualquer outro vértice considerado nos casos seguintes como não pertencentes à F_1, já que Alice deve sempre responder em F_1 quando Bob joga nessa componente do grafo G pelo Lema 4.3. Dito isso, temos os seguintes casos:

1. se $u \notin N(y)$, então Alice irá colorir o vértice y com a cor 2. Agora, para que Alice garanta que todas as cores em $\{1, 2, ..., 2\beta - 1\}$ apareçam em todas as cliques mestre,

é necessário primeiro garantir que a cor 1 apareça em todas as cliques mestre. Como cada clique mestre i possui os vértices de $\{\ell'_{i,0}, \ell''_{i,0}\}$ vizinhos a y, se Bob colorir um vértice ℓ de $\{\ell'_{i,0}, \ell''_{i,0}\}$ para a clique mestre i então seja w o outro vértice de $\{\ell'_{i,0}, \ell''_{i,0}\}$, então Alice irá colorir w, e caso Bob não tenha usado a cor 1 em ℓ, Alice utiliza a cor 1 em w. Se ela não puder usar a cor 1 para w, isso indica que a cor 1 já está presente na clique mestre i.

2. se $u \in N(y)$, então $u \in \{\ell'_{i,0}, \ell''_{i,0}\}$ para alguma clique mestre i. Alice irá então colorir y com uma cor diferente da cor utilizada em u e diferente de 1. Sem perda de generalidade, suponha que Bob tenha utilizado a cor 2 em u e Alice usou a cor 3 em y. Temos agora dois sub-casos a analisar:

   2.a) se logo após a sua jogada em u, Bob colorir o outro vértice $\{\ell'_{i,0}, \ell''_{i,0}\}$ que não seja u e com uma quarta cor. Então Alice deve imediatamente colorir com a cor 1 algum vértice da clique mestre i que seja vizinho de vértices variável, garantindo assim que a cor 1 apareça na clique mestre i. Para as jogadas consecutivas, se Bob colorir um vértice ℓ de $\{\ell'_{j,0}, \ell''_{j,0}\}$ para alguma clique mestre $j \neq i$ então seja w o outro vértice de $\{\ell'_{j,0}, \ell''_{j,0}\}$, então Alice irá colorir w, e caso Bob não tenha usado a cor 1 em ℓ, Alice utiliza a cor 1 em w. Se ela não puder usar a cor 1 para w, isso indica que a cor 1 já está presente na clique mestre j;

   2.b) se logo após a sua jogada em u, Bob não colorir o outro vértice de $\{\ell'_{i,0}, \ell''_{i,0}\}$ que não seja u, e sim um outro vértice de G, então Alice irá colorir o vértice de $\{\ell'_{i,0}, \ell''_{i,0}\}$ que ainda não foi colorido e o fará com a cor 1, caso Bob não já tenha colorido algum vértice da clique mestre i com a cor 1. Após isso, Alice irá gastar todas as suas jogadas seguintes colorindo com a cor 1 um vértice de cada clique mestre até que todas as cliques mestre possuam pelo menos um vértice com a cor 1.

Como cada clique mestre j já possui pelo menos um vértice com a cor 1 e L_j tem pelo menos $4N$ vértices, então qualquer cor que apareça em qualquer um dos N vértices variável pode também aparecer em L_j e os vértices restantes da clique mestre j pode ser propriamente coloridos com o restante das $2\beta - 1$ cores.

Assuma agora que Bob irá colorir o vértice y com a cor 1 em sua primeira jogada. Com isso, Alice segue a seguinte estratégia, sempre em resposta com a última jogada que Bob fez:

1. se Bob joga em um vértice de F_1 então ela joga também em F_1 seguindo sua estratégia vencedora do Lema 4.3;

2. se Bob joga em algum gêmeo-verdadeiro obtido do vértice $\ell_{i,j}$ então Alice joga a menor cor disponível no outro gêmeo-verdadeiro;

3. se Bob joga em algum vértice de L_j então Alice joga como se Bob tivesse passado a vez no POS CNF, o que significa que ela escolhe um vértice de x_j e utiliza uma cor diferente de 1 para colori-lo, onde X_j é o literal escolhido por ela em sua estratégia vencedora no POS CNF;

4. se Bob joga em x_i, então Alice joga como se Bob tivesse escolhido X_i Falso no POS CNF e dessa forma ela escolhe algum vértice x_j para colorir com uma cor diferente 1, onde X_j é literal escolhido por ela em sua estratégia vencedora no POS CNF;

5. se as jogadas anteriores não ocorrem ou não são possíveis, então Alice escolhe qualquer vértice de G e usa a menor cor disponível.

Seguindo tal estratégia, quando o jogo de POS CNF termina, cada clique mestre possui alguns vizinhos exteriores com cor diferente de 1 ou um vértice colorido com a cor 1. Dessa forma, através de um processo de exaustão, Alice e Bob podem terminar de colorir cada clique mestre usando as cores $1, \ldots, 2\beta - 1$, já que cada clique L_j possui pelo menos $4N$ vértices, o que possibilita a utilização de qualquer cor usada nos vértices variável e qualquer outra cor ainda não utilizada para colorir a clique mestre j. Então $\chi_g(G) = \chi(G) = 2\beta - 1$. □

4.2 Complexidade da versão gulosa

Iremos primeiramente propor o seguinte resultado inédito onde caracterizamos grafos G que possuem $\Gamma_g(G) = 2$.

Lema 4.4. *Um grafo conexo G possui $\Gamma_g(G) = 2$ se e somente se G é bipartido com um vértice cuja vizinhança é uma das partes da bipartição.*

Demonstração. Se $\Gamma_g(G) = 2$, então G deve ser bipartido pois $\chi(G) \leq \Gamma_g(G) = 2$. Além disso, se G não é bipartido, então $\Gamma_g(G) \geq \chi(G) > 2$. Então assuma que G é bipartido com bipartição (A, B).

Sem perda de generalidade, suponha que G possui um vértice v de A cuja vizinhança é B. Se Alice colore v em seu primeiro turno (cor 1), então nenhum vértice de B pode ter a cor 1 e consequentemente todos os vértices de A serão coloridos com 1 pelo critério da coloração gulosa e todos os vértices de B terão a cor 2.

Agora suponha que todo vértice de A possui um não-vizinho em B e que todo vértice de B possui um não-vizinho em A. Seja w o primeiro vértice colorido por Alice. Assuma sem perda de generalidade que $w \in A$. Como G é conexo, então w possui um não-vizinho z em B a uma distância 3 (isto é, w e z são extremidades de um P_4). Então, colorindo z em seu primeiro turno (cor 1), Bob garante o uso de pelo menos 3 cores nesse P_4. □

Diferente da versão clássica onde possuímos uma paleta fixa de cores distintas que podem ser usadas durante o jogo de coloração, a versão gulosa possui um "limite" de cores que podem aparecer. Fixado a quantidade máxima de cores que podem aparecer durante uma partida do jogo de coloração gulosa, a dificuldade é determinar se Alice possui uma estratégia vencedora onde não mais que o limite de cores apareçam no jogo.

Uma vez que Alice possui uma estratégia vencedora onde no máximo k cores distintas são usadas durante o jogo, então por definição mais cores não são necessárias para garantir a vitória.

Podemos então definir os seguintes problemas de decisão relacionados ao parâmetro $\Gamma_g(G)$, lembrando que o parâmetro indica a menor quantidade de cores que Alice pode garantir uma estratégia em que não mais do que k cores distintas são usadas durante o jogo de coloração gulosa no grafo G. Dados um grafo G e um inteiro k,

- (Problema 1 do Jogo de Coloração Gulosa) $\Gamma_g(G) \leq k$? Isto é, Alice possui uma estratégia em que não mais do que k cores distintas são usadas durante o jogo de coloração gulosa em G?

Provaremos que o seguinte Problema 2 mais restrito é PSPACE-Completo. Dado um grafo G,

- (Problema 2 do Jogo de Coloração Gulosa) $\Gamma_g(G) = \chi(G)$? Isto é, Alice possui uma estratégia em que não mais do que $\chi(G)$ cores distintas são usadas durante o jogo de coloração gulosa em G?

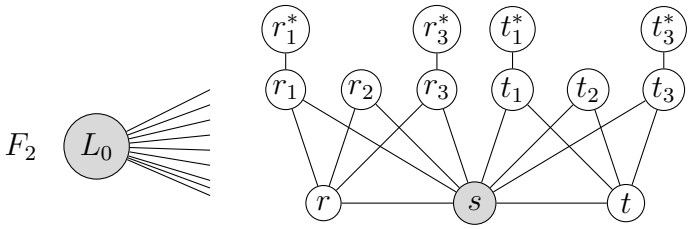


Figura 4.5: Grafo F_2. L_0 é uma clique de tamanho x tal que $N[L_0] = V(F_2)$, isto é, todos os vértices de L_0 são vizinhos de todos os vértices de F_2. Com $3 + x$ cores, o primeiro a jogar ganha o jogo.

É fácil ver que o Problema 1 é uma generalização do Problema 2 (basta fazer $k = \chi(G)$). Então a prova de que o Problema 2 é PSPACE-difícil implica em que o Problema 1 também é PSPACE-difícil.

Obtemos uma redução do problema POS CNF, assim como na versão clássica. Uma parte importante da redução é o grafo F_2 da Figura 4.5. Antes de aprofundarmos no grafo F_2, iremos provar o seguinte lema.

Lema 4.5. *Seja G um grafo parcialmente k-colorido e k um número inteiro. Se G possui um P_4 induzido de vértices v_1, v_2, v_3, v_4 tais que v_1 e v_4 sejam coloridos com a mesma cor, e v_2 e v_3 sejam não coloridos. Então pelo menos 3 cores são necessárias para colorir propriamente G.*

Demonstração. Como v_3 e v_4 possuem vizinhos com a mesma cor, essa cor é proibida para ambos, e no momento que um vértice de $\{v_3, v_4\}$ é colorido, o outro não pode receber a mesma cor pois a coloração deve ser própria. Dessa forma, pelo menos 3 cores são necessárias para terminar de colorir propriamente G. □

Iremos agora mostrar que Alice ganha o jogo em F_2 se e somente se ela é a primeira a jogar e, se for o caso, ela deve colorir o vértice s primeiro. O inteiro x é a quantidade de vértices que a clique L_0 possui.

Lema 4.6. *Alice consegue garantir que não mais do que $x + 3$ cores (onde x é o tamanho da clique L_0) apareçam no grafo F_2 da Figura 4.5 no jogo de coloração gulosa se e somente se ela é a primeira a jogar. Neste caso, ela deve colorir o vértice s primeiro. Além disso, uma estratégia vencedora existe mesmo se Bob puder passar a vez em sua turno.*

Demonstração. Denotaremos por L_1 todos os vértices de F_2 que não aparecem em L_0. Primeiramente note que as cores que eventualmente aparecerem em L_0 não poderão aparecer

em L_1 pois todo vértice de L_0 é vizinho de todos vértice de L_1 e os jogadores devem sempre escolher a menor cor disponível. Então seja x a quantidade de cores usadas em L_0 e y a quantidade de cores distintas usadas em L_1. Iremos mostrar que y é no máximo 3 se Alice começa colorindo o vértice s.

Iremos inicialmente analisar os casos em que Alice é a primeira a jogar e ela joga em um vértice diferente de s em sua primeira jogada no grafo F_2.

Suponha que Alice é a primeira a jogar e ela irá colorir o vértice r primeiro (cor 1 no jogo de coloração gulosa). Então Bob pode colorir o vértice t_2 (cor 1). Alice irá em seguida jogar em algum vértice u e temos os seguintes casos:

- i) se $u \in \{s, t\}$ (cor 2), então em resposta Bob irá colorir t_1^* (cor 1) e agora os vértices t_1 e t (caso $u = s$) ou $\{t_1, s\}$ (caso $u = t$) ambos possuem vizinhos com as cores 1 e 2 e no momento que um for colorido, o outro terá vizinhos com as cores 1, 2 e 3, então precisamos de mais do que $3 + x$ cores para terminar de colorir propriamente G.

- ii) se $u \in \{t_1, t_3, t_1^*, t_3^*\}$ (cor 1), então em resposta Bob irá seguir a seguinte estratégia:

  para $u = t_1$, Bob irá colorir t_3^* (cor 1);

  para $u = t_3$, Bob irá colorir t_1^* (cor 1);

  para $u = t_1^*$, Bob irá colorir t_3 (cor 1);

  para $u = t_3^*$, Bob irá colorir t_1 (cor 1);

  Para todos os quatro vértices analisados acima, após a jogada de Bob, o grafo F_2 terá uma clique de tamanho 3 em que nenhum vértice da clique pode ser colorido com a cor 1, dessa forma necessitamos de 3 cores distintas e diferentes de 1 para terminar de colorir a clique. Então precisamos novamente de mais do que $3+x$ cores para terminar de colorir propriamente F_2.

- iii) se u não é nenhum dos vértices anteriores, então Bob irá colorir o vértice t com a menor cor disponível i (onde i pode ser 2 caso u não seja vizinho de t e 3 caso $u \in L_0$). Dessa forma, o vértice r tem que ser colorido com uma cor diferente de 1 e de i.

  iii a) se em sua jogada seguinte Alice não colorir algum vértice de $\{s, t_1, t_3, t_1^*, t_3^*\}$, Bob irá colorir o vértice s com a menor cor disponível j (diferente de 1 e de i) e em sua jogada seguinte pode colorir t_1^* ou t_3^* (cor 1) e dessa forma fazer com que exista um vértice dentre os vértices não coloridos de L_1 que possui três vizinhos em L_1 com

cores distintas. Então precisamos de mais do que $3 + x$ cores para terminar de colorir propriamente F_2.

iii b) se em sua jogada seguinte Alice colorir s, Bob responde colorindo t_1^* (cor 1) e dessa forma temos um vértice em L_1 ($\{t_1\}$) com vizinhos em L_1 com três cores distintas, então precisamos novamente de mais do que $3 + x$ cores para terminar de colorir propriamente G.

iii c) se em sua jogada seguinte Alice colorir algum vértice $v \in \{t_1, t_3, t_1^*, t_3^*\}$ (cor 1), então em resposta Bob irá seguir a seguinte estratégia:

para $v = t_1$, Bob irá colorir t_3^* (cor 1);

para $v = t_3$, Bob irá colorir t_1^* (cor 1);

para $v = t_1^*$, Bob irá colorir t_3 (cor 1);

para $v = t_3^*$, Bob irá colorir t_1 (cor 1);

Em todos os casos acima, os vértices não coloridos em $\{t_1, s, t_3\}$ são vizinhos e possuem vértices em sua vizinhança em L_1 com as cores 1 e i, dessa forma uma cor j (diferente 1 e i) deve ser usada para colorir os vértices não coloridos em $\{t_1, s, t_3\}$ e no momento que um deles for colorido, teremos um vértice em $\{t_1, s, t_3\}$ que possui vizinhos em L_1 com três cores distintas, então mais uma vez precisamos de mais do que $3 + x$ cores para terminar de colorir propriamente F_2.

Por simetria podemos fazer a mesma argumentação dos casos acima se Alice fizer a sua primeira jogada em t. Considere agora que Alice faça a sua primeira jogada em algum vértice de L_0 (cor 1). Como todo vértice de L_0 é vizinho de todo vértice de L_1, não podemos mais colorir nenhum vértice de L_1 com a cor 1. A estratégia de Bob será primeiramente colorir o vértice t_2 (cor 2). Seja u o vértice que Alice joga em seguida:

- i) se $u \in L_0$ (cor 3), Bob responde colorindo r (cor 2) e temos um caso análogo a um já analisado nesta prova.

- ii) se $u = s$ (cor 3), Bob responde colorindo t_3^* (cor 2) e como os vértices t e t_2 ambos não podem ser coloridos com 2 e 4 e ambos são vizinhos, no momento que um for colorido, o outro não pode receber a mesma cor e teremos que utilizar pelo menos 4 cores distintas nos vértices de L_1, então Alice não consegue garantir no máximo $x + 3$ cores em F_2.

- iii) se $u = r$ (cor 2), Bob responde colorindo t (cor 3). Seja agora v o vértice que Alice joga em seguida:

  para $v \in L_0 \cup \{r_1^*, r_1, r_2, r_3, r_3^*\}$, Bob irá colorir t_3^* (cor 2) onde temos um caso análogo a um já analisado;

  para $v = t_3$ (cor 2), Bob irá colorir t_1^* (cor 2) que é um caso análogo a um já analisado;

  para $v = t_1^*$ (cor 2), Bob irá colorir t_3 (cor 2) e temos novamente um caso análogo a um já analisado;

  para $v = t_3^*$ (cor 2), Bob irá colorir t_1 (cor 2) e temos mais uma vez um caso análogo a um já analisado;

  para $v \in \{s, t\}$ (cor 3), Bob irá colorir t_3^* (cor 2) e temos mais uma vez um caso análogo a um já analisado;

Resta apenas analisar os casos em Alice faz a primeira jogada em $\{t_1^*, t_1, t_2, t_3, t_3^*\}$ (que por simetria são análogos aos casos $\{r_1^*, r_1, r_2, r_3, r_3^*\}$). Se Alice jogar primeiro no vértice t_2 Bob em resposta irá jogar no vértice r (cor 1) e note que esse caso é análogo a quando Alice joga primeiro em r e Bob responde em t_2 e já sabemos que nesse cenário Bob consegue garantir pelo menos 4 cores distintas entre os vértices de L_1.

Se Alice jogar primeiro em t_1 ou t_3, Bob em resposta pode colorir, respectivamente, t_3^* e t_1^* e dessa forma o vértice t_3 (t_1, respectivamente) não pode mais ser colorido com 1, podendo receber uma cor 2 ou 3 nas próximas jogadas. Se em sua jogada seguinte Alice não colorir um vértice de $\{t_2, t_3, s, t\}$ ($\{t_1, t_2, s, t\}$, respectivamente) Bob irá em resposta colorir o vértice t (cor 2) e agora o vértice s possui vizinhos com as cores 1 e 2 e o vértice t_3 (t_1, respectivamente) também, e como os dois são vizinhos, um receberá a cor 3 e o outro a cor 4, então Alice não consegue garantir no máximo 3 cores entre os vértices de L_1.

Alice deve então em sua segunda jogada colorir algum vértice de $\{t_2, t_3, s, t\}$ ($\{t_1, t_2, s, t\}$, respectivamente). Se ela colorir t_2 (cor 1), Bob vai colorir s (cor 2) e agora t_3 e t (t_1 e t, respectivamente) possuem vizinhos com as cores 1 e 2, e como ambos são vizinhos, um vai receber a cor 3 e o outro a cor 4. Se em sua segunda jogada ela colorir o vértice t_3 (t_1, respectivamente) usando a cor 2, Bob irá colorir s com a cor 3 e agora t possui 3 vizinhos com cores distintas. Se em sua segunda jogada ela colorir o vértice s (cor 2), Bob irá colorir t_3 (t_1, respectivamente) com a cor 2, garantindo que pelo menos um vértice de $\{t_2, t\}$ seja

colorido com a cor 4 e o caso em que Alice faz a sua segunda jogada em t é análogo ao caso em que ela faz sua segunda jogada em s.

Iremos mostrar agora que Alice consegue garantir que no máximo três cores distintas aparecem entre os vértices de L_1 se ela jogar primeiro em s.

No começo ela irá colorir o vértice s (cor 1). Agora os vértices $\{r_1^*, r_3^*, t_1^*, t_3^*\}$ só podem ser coloridos com a cor 1 até o fim do jogo e os vértices restantes só receberão no máximo duas cores distintas entre si, mesmo se Bob jogar em L_0 ou passar a sua vez, pois os subconjuntos de vértices $\{r, r_1, r_2, r_3\}$ e $\{t, t_1, t_2, t_3\}$ ambos induzem um grafo bipartido completo $K_{1,3}$, e quando uma cor é usada em uma das partes de um grafo bipartido completo, ela não pode ser utilizada na outra parte e no momento que uma cor é usada em uma das partições, ela sempre pode ser usada nessa partição.

Finalmente, suponha que Bob é o primeiro a jogar. Ele pode garantir que quatro cores distintas apareçam entre os vértices de L_1 se ele começar colorindo o vértice r_2 primeiro (cor 1). Note que o caso em que Bob joga primeiro é similar ao caso em que Alice joga primeiro e joga em um vértice de L_0, o que já analisamos e sabemos que Bob garante que mais do que três cores distintas entre os vértices de L_1, então Alice consegue garantir no máximo $x + 3$ cores caso ela jogar primeiro e Bob garante mais do que $x + 3$ cores caso ele jogar primeiro. □

Utilizamos o grafo F_2 e o Lema 4.6 para auxiliar a prova de que decidir se $\Gamma_g(G) \leq k$ para um k inteiro é um problema PSPACE-Completo. Iremos mostrar que para uma fórmula F onde um jogo de POS CNF é jogado existe um grafo G tal que Alice possui uma estratégia vencedora para o jogo de coloração gulosa em G se e somente se ela também possui uma estratégia vencedora em F.

Teorema 4.2. *Dado um grafo G, decidir se $\Gamma_g(G) = \chi(G)$ é PSPACE-Completo. consequentemente, se k é um inteiro, decidir se $\Gamma_g(G) \leq k$ é um problema PSPACE-Completo.*

Demonstração. Pelo Lema 4.1 o problema pertence à PSPACE, resta mostrar que o problema é PSPACE-difícil. Iremos seguir uma redução semelhante à do Teorema 4.1, assim como a terminologia utilizada na prova, partindo também do POS CNF. A construção do grafo G também é semelhante, mudando o grafo F_1 pelo grafo F_2 onde L_0 é uma clique com $2\beta - 4$ vértices, e para cada vértice x_i iremos adicionar um vizinho $\overline{x_i}$ de grau 1 e para o vértice y iremos também adicionar um vizinho $\overline{y}$ de grau 1. Também nesse caso temos que

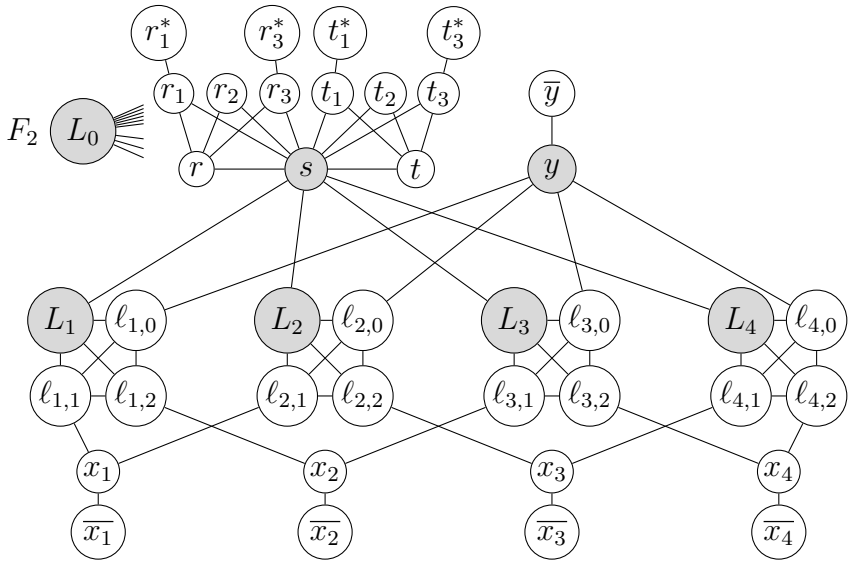


Figura 4.6: Grafo construído G para a fórmula $(X_1 \vee X_2) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_4) \wedge (X_3 \vee X_4)$. Relembre que cada vértice $\ell_{j,k}$ representa dois gêmeos-verdadeiros $\ell'_{j,k}$ e $\ell''_{j,k}$, L_0 é uma clique com 9 vértices e L_1, L_2, L_3, L_4 são cliques com 6 vértices. Bob possui uma estratégia vencedora com 12 cores no jogo de coloração gulosa.

L_j é uma clique com $2(\beta - p_j) - 3$ vértices, para $1 \leq j \leq M$ e cada clique mestre i possui $2\beta - 1$ vértices.

A Figura 4.6 mostra o grafo construído G para a mesma fórmula $(X_1 \vee X_2) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_4) \wedge (X_3 \vee X_4)$. Na redução desse exemplo, temos os mesmos valores $N = 4$ variáveis, $M = 4$ cláusulas, $p = 2$, $q = 2$, $\beta = 4N = 16$, a clique L_0 possui $2\beta - 4 = 28$ vértices e as cliques L_1 à L_M com $2(\beta - p) - 3 = 25$ vértices cada e 31 vértices em cada clique mestre i.

Assim como no Teorema 4.1, é fácil ver que $\chi(G) = 2\beta - 1$. Iremos mostrar agora que Alice possui uma estratégia vencedora para o jogo de coloração gulosa se e somente se ela possui uma estratégia vencedora no POS CNF. Pelo Lema 4.6, em sua primeira jogada Alice deve colorir o vértice s de F_2, caso contrário F_2 não pode ser colorido com $2\beta - 1$ cores.

Note que cada vértice de uma clique cláusula possui grau exatamente $2\beta - 1$. Para garantir que todas as cores de $\{1, ..., 2\beta - 1\}$ possam ser usadas em todos os vértices de G, Alice deve garantir que todas as cores que apareçam na vizinhança externa de cada clique cláusula também apareça dentro dela. Também iremos mostrar que, assim como na redução da versão clássica do jogo, a estratégia de Bob é fazer com que toda a vizinhança exterior de pelo menos uma clique mestre tenham a mesma cor, impossibilitando que Alice possa usar essa cor dentro da clique.

Para o jogo de coloração gulosa e para o grafo construído G, colorir um vértice x_i com

a cor 1 significa marcar o literal X_i do POS CNF como Falso e colorir um vértice x_j com uma cor diferente de 1 significa marcar o literal X_j do POS CNF como Verdadeiro.

Primeiramente mostramos que se Bob possui uma estratégia vencedora no jogo de POS CNF, então $\Gamma_g(G) > 2\beta - 1$. Assuma que Bob vence no POS CNF e Alice começa colorindo o vértice s com a cor 1 (Lema 4.6)). Bob usará a seguinte estratégia. Em seu primeiro turno ele irá colorir y com a cor 1, e então para cada turno subsequente: se Alice colorir algum vértice de $N[x_i]$ (vizinhança fechada de x_i) para algum i, Bob considera que ela marcou X_i como verdadeiro no jogo de POS CNF e com a cor 1 irá colorir o vértice x_j representando o literal X_j escolhido por ele em sua estratégia vencedora do POS CNF; se Alice não colorir qualquer vértice de $N[x_i]$ para algum i, então Bob joga como se Alice tivesse passado seu turno no POS CNF. Como Bob possui uma estratégia vencedora no jogo de POS CNF, em algum ponto todos os literais de alguma cláusula estarão marcados como Falsos, o que significa que para o jogo de coloração gulosa existe alguma clique cláusula com toda a vizinhança externa com a cor 1. Então $\Gamma_g(G) > 2\beta - 1$.

Mostramos agora que se Alice possui uma estratégia vencedora no jogo de POS CNF então $\Gamma_g(G) = 2\beta - 1$. Assuma que Alice ganha no jogo de POS CNF e que ela irá colorir o vértice s de F_2 com a cor 1 em sua primeira jogada.

Suponha primeiramente que Bob irá colorir um vértice u diferente de y em seu primeiro turno. Iremos supor que y e qualquer outro vértice considerado nos casos seguintes como não pertencentes à F_2, já que Alice deve sempre responder em F_2 quando Bob joga nessa componente do grafo G pelo Lema 4.6. Dito isso, temos os seguintes casos:

- i) se $u \notin \{y, \overline{y}\}$, então Alice irá colorir o vértice $\overline{y}$ com a cor 1 (coloração gulosa). Como o vértice y agora pode receber uma cor diferente de 1, então os vértices em sua vizinhança ($\{\ell'_{i,0}, \ell''_{i,0}\}$ para cada clique mestre i) só não poderão ser coloridos com 1 caso já exista um vértice na clique mestre i com a cor 1.

- ii) se $u \in N(y) - \{\overline{y}\}$, então $u \in \{\ell'_{i,0}, \ell''_{i,0}\}$ para alguma clique mestre i. Porém como a coloração é gulosa, u recebe a cor 1 e y receberá uma cor diferente de 1 e poderemos colorir pelo menos um dos vértices de $\{\ell'_{i,0}, \ell''_{i,0}\}$ com a cor 1.

Como cada clique mestre j já possui pelo menos um vértice com a cor 1 e L_j tem pelo menos $4N$ vértices, então qualquer cor que apareça em qualquer um dos N vértices variável pode também aparecer em L_j e os vértices restantes da clique mestre j pode ser propriamente coloridos com o restante das $2\beta - 1$ cores.

Assuma então que Bob irá colorir o vértice y (cor 1). Com isso, Alice pode jogar usando a seguinte estratégia:

- 1) se Bob joga em algum vértice de F_2 e ainda existem vértices a serem coloridos em F_2, então ela também joga em F_2 seguindo sua estratégia vencedora do Lema 4.6;
- 2) se Bob joga em algum gêmeo-verdadeiro obtido do vértice $\ell_{i,j}$ então Alice joga no outro gêmeo-verdadeiro;
- 3) se Bob joga em algum vértice de L_j, em $\overline{y}$ ou joga em F_2, terminando de colori-lo, então Alice joga como se Bob tivesse passado a vez no jogo de POS CNF, o que significa que ela escolhe um vértice de $\overline{x_j}$ e a cor 1 para colori-lo, onde X_j é o literal por ela em sua estratégia vencedora no POS CNF;
- 4) se Bob joga em x_i, então Alice joga como se Bob tivesse escolhido X_i Falso no POS CNF;
- 5) se as jogadas anteriores não ocorrem, então Alice escolhe qualquer vértice de G para colorir.

Assim como no Teorema 4.1, ao seguir a estratégia descrita, quando o jogo de POS CNF termina, cada clique mestre possui alguns vizinhos exteriores com cor diferente de 1 ou um vértice colorido com a cor 1. Dessa forma, Alice e Bob podem terminar de colorir cada clique mestre usando as cores $1, \ldots, 2\beta - 1$, pois Alice pode garantir que cada uma das cores que aparecerem nos vértices x_i apareçam em vértices de L_j pois como cada L_j tem pelo menos $4N$ vértices e temos que N vértices x_i, mesmo se todas as cores forem distintas, ainda temos como colorir vértices de L_j com todas as cores.

□

5 NÚMERO DE GRUNDY LÚDICO

Em 2013 Havet & Zhu mostraram em (HAVET; ZHU, 2013) que $\Gamma_g(G) = \chi(G)$ para qualquer cografo (veja a Proposição 9 do artigo), já que $\chi(G) = \Gamma(G)$ em cografos. Neste capítulo, iremos provar que isso também ocorre em superclasses de cografos, como é o caso de P_4-esparsos, P_4-tidy e P_4-laden, de tal forma que $\Gamma(G)$ possa ser maior que $\chi(G)$ o quanto for necessário.

Um *cografo* é um grafo sem P_4 induzido (veja (CORNEIL *et al.,* 1981)). Um grafo G é P_4-*esparso* se todo conjunto de cinco vértices em G induz no máximo um P_4 (JAMISON; OLARIU, 1992). Grafos P_4-esparsos possuem uma boa estrutura em termos de uniões e junções. Dados grafos $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$, a *união de* G_1 *e* G_2 é o grafo $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$ e a *junção de* G_1 *e* G_2 é o grafo $G_1 \vee G_2 = (V_1 \cup V_2, E_1 \cup E_2 \cup \{uv : u \in V_1, v \in V_2\})$.

Uma *aranha* é um grafo cujo conjunto de vértices possui a partição (R, C, S), onde $C = \{c_1, \ldots, c_k\}$ e $S = \{s_1, \ldots, s_k\}$ (com $k \geq 2$) são respectivamente uma clique e um conjunto independente; s_i é adjacente à c_j se e somente $i = j$ (uma aranha magra), ou s_i é adjacente à c_j se e somente se $i \neq j$ (uma aranha gorda); e todo vértice de R é adjacente a cada vértice de C e não-adjacente a cada vértice de S. Note que o complemento de uma aranha magra é uma aranha gorda, e vice-versa.

Jamison e Olariu em (JAMISON; OLARIU, 1992) provaram que, se um grafo G é P_4-esparso, então G é a união disjunta ou a união de dois grafos P_4-esparso, ou G é uma aranha (R, C, S) tal que $G[R]$ é um grafo P_4-esparso, ou G tem no máximo um vértice.

Como um exemplo, a seguinte sequência (G_k) de grafos P_4-esparsos satisfaz $\Gamma_g(G_k) = \chi(G_k) = 2k$ e $\Gamma(G_k) = 3k$. Seja G_1 um P_4 $a_1b_1c_1d_1$. Para $k \geq 2$, seja G_k obtido da junção de G_{k-1} com um P_4 $a_kb_kc_kd_k$. Claramente, $\chi(G_k) = 2k$: ponha em a_i e c_i a cor $2i - 1$ e ponha em b_i e d_i a cor $2i$ para cada $1 \leq i \leq k$. Adicionalmente, $\Gamma(G_k) = 3k$: ponha em a_i e d_i a cor $3i - 2$ e em b_i e c_i as cores $3i - 1$ e $3i$, respectivamente. Finalmente, $\Gamma_g(G_k) = 2k$: Alice pode sempre evitar que ambas as extremidades a_i e d_i de um P_4 $a_ib_ic_id_i$ recebam a mesma cor.

Dizemos que um grafo é P_4-*laden estendido* se todo subgrafo induzido com no máximo seis vértices que contém dois uma mais P_4 induzidos é livre de $\{2K_2, C_4\}$. Essa classe de grafo foi introduzida em (GIAKOUMAKIS, 1996), e uma motivação para o desenvolvimento

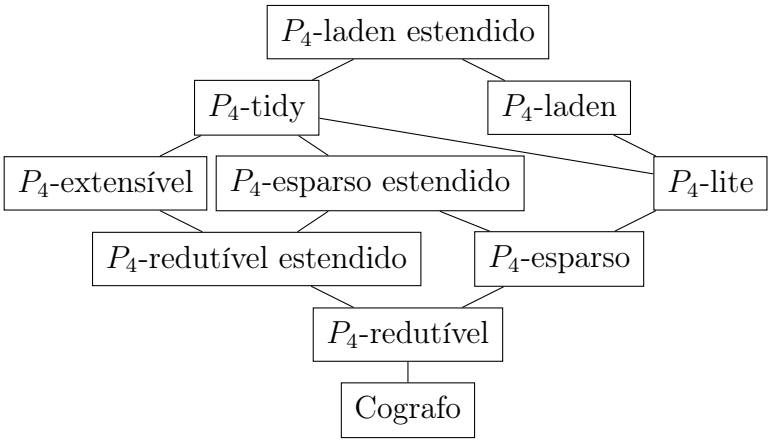


Figura 5.1: Hierarquia de grafos com poucos P_4's.

de algoritmos para P_4-laden estendidos segue do fato que eles estão no topo de uma hierarquia de classes vastamente estudada que contém várias grafos com poucos P_4's (veja Figura 5.1), incluindo cografos, P_4-esparsos, P_4-lites, P_4-ladens e P_4-tidys. Portanto, resolver problemas interessantes de forma eficaz para grafos P_4-laden imediatamente implica algoritmos eficientes generalizados para todas essas classes de grafos.

Uma outra motivação é a de que grafos P_4-laden estendidos não estão contidos em grafos perfeitos; logo os seguintes resultados obtém resultados em coloração não especificamente relacionados à perfeição em grafos. Existem trabalhos recentes que lidam com grafos P_4-laden estendidos. Por exemplo, no número de Grundy (ARAUJO; SALES, 2012) e no número cocromático (BRAVO *et al.,* 2012).

Giakoumakis em (GIAKOUMAKIS, 1996) provou uma importante caracterização estrutural para grafos P_4-laden estendido, chamados *pseudo-splits* e *quasi-aranhas*. Um grafo *split* é um grafo G com partição de vértices (C, S), onde C é uma clique e S é um conjunto independente. Um grafo split G é *original* se todo vértice em S possui um não-vizinho em C e todo vértice em C possui um vizinho em S. Dizemos que um grafo G é um *pseudo-split* se seu conjunto de vértices possui uma partição (R, C, S) tal que S induz um conjunto independente, C induz uma clique, $C \cup S$ induz um grafo split original e todo vértice de R é adjacente a todo vértice de C e não-vizinho a todo vértice de S. Note que o complemento de um grafo pseudo-split é um também um pseudo-split e que aranhas são grafos pseudo-splits. Uma *quasi-aranha* é um grafo obtido de uma aranha (R, C, S) com no máximo um vértice de $C \cup S$ substituído por K_2 ou $\overline{K_2}$ (mantendo a vizinhança). Claramente, toda aranha é uma quasi-aranha.

Teorema 5.1 ((GIAKOUMAKIS, 1996)). *Um grafo G é um P_4-laden estendido se e somente se exatamente uma das seguintes situações ocorre:*

(a) G é a união disjunta ou a junção de dois grafos P_4-laden estendidos não-vazios.

(b) G é uma quasi-aranha ou um pseudo-split (R, C, S) tal que $G[R]$ é um grafo P_4-laden estendido.

(c) G é isomorfo à C_5, P_5, $\overline{P_5}$, ou possui no máximo um vértice.

Para determinar $\Gamma_g(G)$ para grafos P_4-laden estendido, introduzimos o parâmetro $\Gamma'_g(G)$: menor quantidade de cores tal que Alice possui uma estratégia vencedora para o jogo de coloração gulosa mesmo se Bob pode começar o jogo e pode passar a sua vez (isto é, não colorir um vértice em seu turno). Note que $\Gamma_g(G) \leq \Gamma'_g(G)$ pois Bob possui uma quantidade maior de jogadas possíveis em seu turno pois ele também pode passar sua vez, enquanto Alice retém a mesma quantidade de possibilidades em sua vez.

Sabemos que $\chi(G) \leq \Gamma_g(G) \leq \Gamma(G)$ para qualquer grafo G (Capítulo 3.2), então se mostrarmos que existem grafos G tais que $\Gamma'_g(G) \leq \chi(G)$ então também existem grafos G com $\Gamma_g(G) = \chi(G)$.

Iremos provar que $\Gamma'_g(G) = \chi(G)$ para grafos P_4-laden estendido, implicando que $\Gamma_g(G) = \chi(G)$ nessa classe de grafos. Iremos propor vários lemas que irão auxiliar na prova do resultado principal deste capítulo.

Lema 5.1. *Dados grafos G_1 e G_2,*

- $\Gamma'_g(G_1 \vee G_2) \leq \Gamma'_g(G_1) + \Gamma'_g(G_2)$ *e*
- $\Gamma'_g(G_1 \cup G_2) \leq \max\{\Gamma'_g(G_1), \Gamma'_g(G_2)\}$.

Demonstração. Para a junção e união, basta Alice evitar que Bob jogue duas vezes seguidas na mesma componente. Como Alice garante no máximo $\Gamma'_g(G_1)$ cores distintas em G_1 e no máximo $\Gamma'_g(G_2)$ cores distintas em G_2 mesmo quando Bob pode passar a sua vez, então basta que ela jogue na componente que Bob jogou por último e utilize a sua estratégia vencedora, e se Bob passar a sua vez basta ela colorir algum vértice de G_1 ou G_2 ainda não colorido utilizando a sua estratégia vencedora. Como todo vértice de G_1 é vizinho de todo vértice de G_2 no grafo $G_1 \vee G_2$, então Alice garante no máximo $\Gamma'_g(G_1) + \Gamma'_g(G_2)$ cores para a junção.

Para a união, Alice novamente joga sempre na última componente que Bob jogou, usando sua estratégia vencedora. Como Alice garante no máximo $\Gamma'(G)$ em qualquer grafo G no

jogo de coloração gulosa na versão em que Bob pode passar a sua vez e pode ser o primeiro a jogar, mesmo se Alice não puder jogar na última componente que Bob jogou (caso em que a componente está completamente colorida após a jogada de Bob), ela ainda garante $\max\{\Gamma'_g(G_1), \Gamma'_g(G_2)\}$ no grafo $G_1 \cup G_2$ pois nenhum vértice de G_1 é vizinho de qualquer vértice de G_2.

□

Lema 5.2. *Se G é um grafo pseudo-split com partição de vértices (R, C, S), então $\Gamma'_g(G) \leq \Gamma'_g(G[R]) + |C|$.*

Demonstração. Seja G um grafo pseudo-split com partição de vértices (R, C, S). Se $R \neq \emptyset$, então a estratégia de Alice é sempre jogar em R seguindo a sua estratégia vencedora em R. Quando R estiver completamente colorido, ela joga em C. As cores que aparecem em R podem ser usadas em S, pois não existem arestas em grafos pseudo-split onde temos uma extremidade em R e a outra em S, e as cores que aparecem em C são todas distintas e diferentes das cores que aparecem em R, então $\Gamma'_g(G) \leq \Gamma'_g(G[R]) + |C|$ neste caso.

Se $R = \emptyset$, Alice só precisa garantir que os vértices de S não recebam todos a mesma cor. Se Alice começa jogando, basta ela colorir algum vértice de C com a cor 1, e se Bob começa jogando e colorindo algum vértice de S (cor 1), basta Alice colorir o não-vizinho de S em C (que existe pela definição de grafo pseudo-split) com a cor 1, e agora pelo menos um vértice de S não pode receber a cor 1, então $\Gamma'_g(G) \leq \Gamma'_g(G[R]) + |C|$ também neste caso.

□

Lema 5.3. *Se G é uma quasi-aranha com partição de vértices (R, C, S), então :*

- $\Gamma'_g(G) \leq \Gamma'_g(G[R]) + |C| + 1$ *se nenhum vértice de $C \cup S$ é substituído por um K_2 ou $\overline{K_2}$, ou no máximo um vértice de C é substituído por um K_2;*
- $\Gamma'_g(G) \leq \Gamma'_g(G[R]) + |C|$, *caso contrário.*

Demonstração. Seja G uma quasi-aranha, se nenhum vértice de $C \cup S$ é transformado em um K_2 ou $\overline{K_2}$, então a estratégia de Alice é colorir vértices de R com a sua estratégia vencedora em R garantindo $\Gamma'_g(G[R])$ distintas e mesmo se em algum momento do jogo todos os vértices de S forem coloridos com a mesma cor (o que impossibilita o desta cor em C), ainda podemos colorir vértices de C de tal forma que teremos no máximo $|C| + 1$ cores

distintas entre os vértices de $C \cup S$, e como todo vértice de R é vizinho de todos vértice de C, Alice garante no máximo $\Gamma'_g(G[R]) + |C| + 1$ cores distintas em G.

Resta então analisar os casos onde algum vértice de $C \cup S$ é transformado um K_2 ou um $\overline{K_2}$. Temos dois casos principais onde G é construído a partir de uma aranha gorda ou uma aranha magra, em seguida para cada tipo de aranha analisamos qual vértice de $C \cup S$ é transformado em um K_2 ou um $\overline{K_2}$.

- 1) G é construído a partir de uma aranha gorda.

  1 a) Um vértice de C é substituído por um K_2. Nesse caso C é apenas uma clique com um vértice a mais que o anterior e o grafo é um pseudo-split que, pelo Lema 5.2, implica que $\Gamma'_g(G) \leq \Gamma'_g(G[R]) + |C| + 1$, pois adicionamos um vértice à C.

  1 b) Um vértice de C é substituído por um $\overline{K_2}$. Nesse caso, Alice irá jogar sempre em R, até que R esteja completamente colorido, garantindo no máximo $\Gamma'_g(G[R])$ cores distintas entre os vértices de R. Seja w o vértice que foi transformado em um $\overline{K_2}$ sejam w_1 e w_2 os vértices que formam o $\overline{K_2}$, e seja s_i o vértice de S que era não-vizinho de w. Como w_1, w_2 e s_i são todos não-vizinhos entre si, eles podem ser coloridos com a mesma cor. Então teremos ao fim do jogo $|C|$ cores distintas em C onde cada uma dessas $|C|$ cores podem aparecer em S pois existe pelo menos um não vizinho em C para cada vértice em S. Então $\Gamma'_g(G) \leq \Gamma'_g(G[R]) + |C|$.

  1 c) Um vértice de S é substituído por um K_2. Temos que considerar dois subcasos: quando R é vazio e quando R não é vazio.

  1 c.1) $R = \emptyset$. Seja c_i o vértice de C que não possui vizinhos em K_2 e seja $C' = C - \{c_i\}$. Note que o conjunto de vértices $\{C'\} \cup K_2$ induz um K_{n+1} onde n é a quantidade de vértices em C. Alice precisa então garantir que todas as cores que aparecem em K_{n+1} apareçam nos vértices restantes, para isso a primeira jogada de Alice será colorir algum vértice do K_2 (cor 1). Caso algum outro vértice de S seja colorido (cor 1) antes que c_i seja colorido, a cor usado no outro vértice de K_2 pode ser usada em c_i e dessa forma Alice garante que não mais do que $\Gamma'_g(G[R]) + |C| + 1$ cores distintas apareçam em G.

  1 c.2) $R \neq \emptyset$. Alice nesse caso deve evitar que Bob jogue duas vezes em R, caso contrário ela não garante no máximo $\Gamma'_g(G[R])$ cores distintas entre os vértices de R, porém ela também deve evitar que os vértices de $K_2 \cup \{c_i\}$ recebam todos três

cores distintas. A estratégia de Alice será então colorir um dos vértices de K_2 em sua primeira jogada (cor 1). Agora para cada jogada que Bob faz em R, ela responde em R e quando uma jogada em R não for possível (caso Bob jogue em algum vértice fora de R ou R esteja completamente colorido), então Alice irá colorir o outro vértice de K_2 caso ainda não esteja colorido e dessa forma garantindo que c_i receba umas das cores presentes em K_2, garantindo assim não mais do que $\Gamma'_g(G[R]) + |C| + 1$ cores em G.

1 d) Um vértice de S é substituído por um $\overline{K_2}$. Temos que considerar dois subcasos: quando R é vazio e quando R não é vazio.

1 d.1) $R = \emptyset$. Seja c_i o vértice de C que não possui vizinhos em K_2 e seja $C' = C - \{c_i\}$. Alice só precisa garantir que todas as cores que apareçam em S também apareçam em C, para isso ela irá primeiramente colorir c_i (cor 1) e como mais nenhum vértice de C pode ser colorido com 1, os vértices de $\overline{K_2}$ receberão a cor 1 quando forem coloridos e cada vértice restante de S receberá a cor de seu respectivo não-vizinho em C, garantindo então que no máximo $\Gamma'_g(G[R]) + |C|$ cores em G (onde $\Gamma'_g(G[R]) = 0$).

1 d.2) $R \neq \emptyset$. Alice começa colorindo c_i (cor 1) e agora sempre que Bob jogar em R ela responde em R e se ela não puder (caso R esteja completamente colorido) ela joga em qualquer outro vértice. Como c_i possui a cor 1, os vértices de $\overline{K_2}$ serão coloridos com 1 e cada vértice de S receberá a cor de seu respectivo não vizinho em C, garantindo assim no máximo $\Gamma'_g(G[R]) + |C|$ cores distintas em G.

- 2) G é construído a partir de uma aranha magra.

2 a) Um vértice de C é substituído por um K_2. A estratégia é igual à do item 1 a), então Alice garante no máximo $\Gamma'_g(G) \leq \Gamma'_g(G[R]) + |C| + 1$ cores em G.

2 b) Um vértice de C é substituído por um $\overline{K_2}$. Seja c_i o vértice de C que virou um $\overline{K_2}$, sejam $\{w_1, w_2\}$ os vértices do $\overline{K_2}$ e seja w o vértice de S que é vizinho de w_1 e w_2. Alice começa colorindo w_1 com a cor 1 e agora w_2 pode receber a cor 1. Como os vértices restantes de C agora só podem receber no máximo $|C|$ cores distintas entre si e os vértices de S podem receber as mesmas cores de R, que são todas distintas das cores de C, então $\Gamma'_g(G) \leq \Gamma'_g(G[R]) + |C|$.

2 c) Um vértice de S é substituído por um K_2. Seja c_i o vértice de C vizinho à

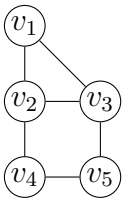


Figura 5.2: Grafo G isomorfo a um $\overline{P_5}$.

K_2. Alice começa o jogo colorindo o vértice c_i (cor 1) e agora mais nenhum vértice de C pode receber a cor 1, o que implica que todos os vértices restantes de S, fora o K_2, receberão a cor 1 e os vértices do K_2 receberão as cores 1 e 2, então a estratégia de Alice é sempre jogar em R até R estar completamente colorindo e após isso colorir qualquer outro vértice em seu turno. Seguindo essa estratégia ela garante não mais do que $\Gamma'_g(G[R]) + |C|$ cores distintas em G.

2 d) Um vértice de S é substituído por um $\overline{K_2}$. A estratégia é a mesma do item 2 c), troque apenas K_2 por $\overline{K_2}$ e os vértices de $\overline{K_2}$ recebem ambos a cor 2.

□

Lema 5.4. *Se um grafo G é isomorfo a um C_5, P_5 ou $\overline{P_5}$, então $\Gamma'_g(G) = \chi(G)$.*

Demonstração. Se G é isomorfo a um C_5, como $\Delta(G) = 2$, então não mais do que três cores distintas aparecem em G pois $\Gamma'(G) \leq \Delta(G) + 1$ e como G é um ciclo ímpar, então $\chi(G) = 3$, então $\Gamma'_g(G) = \chi(G)$.

Se G é isomorfo a um P_5, então basta que Alice jogue no vértice central v do P_5, agora todo vértice vizinho a v recebe a cor 2 e outros recebem a cor 1 e claramente $\chi(P_5) = 2$, então $\Gamma'_g(G) = \chi(G)$.

Se G é isomorfo a um $\overline{P_5}$, sejam v_1, ..., v_5 os vértices de G assim como na Figura 5.2. Alice começa colorindo o vértice v_3 (cor 1) e agora o vértice v_4 só pode receber a cor 1, v_5 só pode receber a cor 2 e os vértices de v_1 e v_2 receberão um a cor 2 e o outro a cor 3. Claramente $\chi(\overline{P_5}) = 3$, então $\Gamma'_g(G) = \chi(G)$. □

Lema 5.5. *Se G_1 e G_2 são grafos da classe P_4-laden estendido, então,*

- $\Gamma'_g(G_1) + \Gamma'_g(G_2) = \chi(G_1) + \chi(G_2) = \chi(G_1 \vee G_2)$ *e*
- $\max\{\Gamma'_g(G_1), \Gamma'_g(G_2)\} = \max\{\chi(G_1), \chi(G_2)\} = \chi(G_1 \cup G_2)$.

Demonstração. Iremos provar por indução no número de vértices em G. Nos casos base temos que G pode ser um C_5, P_5 ou $\overline{P_5}$ ou apenas um vértice isolado. Os três primeiros casos sabemos que são iguais a $\chi(G)$ pelo Lema 5.4 e para o vértice isolado v temos trivialmente $\Gamma'_g(G) = \chi(G)$.

Para ambas união e junção, a estratégia de Alice é sempre jogar na última componente, seja ela G_1 ou G_2 que Bob jogou, sempre usando sua estratégia vencedora na componente, ao fazer isso ela garante não mais que $\Gamma'_g(G)$ em cada uma das componentes e se temos uma junção, as cores presentes em uma componente devem ser diferentes das presentes na outra componente, resultando em $\chi(G_1) + \chi(G_2)$.

Para a união, Alice garante $\chi(G_1)$ e $\chi(G_2)$ em cada componente mas como cada vértice de uma componente é não vizinho da outra, então o máximo de cores usadas em $G_1 \cup G_2$ é $\max\{\chi(G_1), \chi(G_2)\}$.

Claramente $\chi(G_1) + \chi(G_2) = \chi(G_1 \vee G_2)$ e $\max\{\chi(G_1), \chi(G_2)\} = \chi(G_1 \cup G_2)$, então os casos base valem.

Suponha que ambas proposições sejam válidas para grafos com $n < k$ vértices, e seja primeiramente $G = G_1 \cup G_2$ a união de dois grafos P_4-laden estendido não-vazios onde G possui $n = k$ vértices. A estratégia de Alice é começar a jogar em uma das duas componentes e agora sempre que Bob jogar em uma componente G_i, ela responde em G_i com a sua estratégia vencedora. Como cada componente G_i possui uma quantidade de vértices menor do que k, e G_i é um grafo P_4-laden estendido, então G_i é exatamente um dos três seguintes, pelo Teorema 5.1

(a) G_i é a união disjunta ou a junção de dois grafos P_4-laden estendidos não-vazios.

(b) G_i é uma quasi-aranha ou um pseudo-split (R, C, S) tal que $G[R]$ é um grafo P_4-laden estendido.

(c) G_i é isomorfo à C_5, P_5, $\overline{P_5}$, ou possui no máximo um vértice.

Se G_i for o caso (a), temos por hipótese de indução que ambas proposições valem, se G_i for o caso (b) temos que G_i é uma quasi-aranha ou um pseudo-split (R, C, S) tal que $G[R]$ é um grafo P_4-laden estendido que pelos Lemas 5.2 e 5.3 garante não mais do que $\Gamma'_g(G[R]) + |C|$ ou $\Gamma'_g(G[R]) + |C| + 1$ cores distintas por hipótese de indução temos que esse valor é igual à $\chi(G)$ ou $\chi(G) + 1$, e se temos o caso (c) sabem que as proposições são válidas pelos casos base.

Então se as proposições valem para as componentes com $n < k$ vértices, usando a estratégia de Alice sempre responder na última componente que Bob joga ela garante $\Gamma'_g(G_i)$ em cada componente que por hipótese de indução é igual a $\chi(G_i)$, então $\Gamma'_g(G_1) + \Gamma'_g(G_2) = \chi(G_1) + \chi(G_2) = \chi(G_1 \vee G_2)$ e $\max\{\Gamma'_g(G_1), \Gamma'_g(G_2)\} = \max\{\chi(G_1), \chi(G_2)\} = \chi(G_1 \cup G_2)$. □

Teorema 5.2. *Se G é um grafo P_4-laden estendido, então $\Gamma_g(G) = \Gamma'_g(G) = \chi(G)$, que pode ser computado em tempo linear. Isto é, Alice ganha o jogo de coloração gulosa com $\chi(G)$ cores em grafos P_4-laden estendido mesmo se Bob puder começar o jogo e também puder passar sua vez em qualquer jogada.*

Demonstração. Segue diretamente dos Lemas 5.1, 5.2, 5.3, 5.4 e 5.5. □

6 CONCLUSÃO

No Capítulo 4 conseguimos a resposta de uma pergunta da área de jogo de coloração em grafos que permaneceu em aberto por 28 anos: a complexidade do jogo de coloração. Mostramos que o problema de decidir se Alice possui uma estratégia vencedora para a versão clássica do jogo de coloração em um grafo G usando k cores ou menos é PSPACE-Completo. Mostramos também que a versão gulosa do jogo de coloração proposta por Havet também é PSPACE-Completo e propomos uma caracterização de grafos em que Alice precisa de exatamente duas cores para vencer o jogo de coloração gulosa.

No capítulo 5 obtivemos limitantes superiores para o parâmetro $\Gamma_g(G)$ quando G é uma junção ou união de dois grafos. Obtivemos valores exatos de $\Gamma_g(G)$ quando G é um grafo P_4-laden estendido, que é a classe que se encontra no topo de uma hierarquia de classes de grafos com poucos P_4-induzidos.

Os resultados de ambos capítulos 4 e 5 foram aceitos em forma de artigo para o Simpósio Latino Americano de Algoritmos, Grafos e Otimização de 2019 (LAGOS 2019).

A caracterização, assim como a complexidade de grafos que possuem $\Gamma_g(G) \leq 3$ ainda é um problema em aberto. Estudamos também a conjectura de que se Alice possui uma estratégia vencedora com k cores em uma grafo G para a versão clássica do jogo de coloração, então ela também possui uma estratégia vencedora com k' cores, onde $k' > k$. A conjectura é válida para todos os isomorfismos de grafos de até 8 vértices, onde testamos as instâncias (GRAPHS, Acesso em: 29 Dez. 2018).

O código utilizado pode ser encontrado no Apêndice A desta dissertação. O resultado positivo em todos os isomorfismos de grafos com até 8 vértices nos leva a acreditar que a conjectura pode ser válida para qualquer grafo simples. Para trabalhos futuros planejamos encontrar uma prova para a conjectura.

REFERÊNCIAS

ARAUJO, J.; SALES, C. L. On the Grundy number of graphs with few P4's. **Discrete Applied Mathematics.** Elsevier, v. 160, n. 18, p. 2514–2522, 2012.

BARTNICKI, T.; GRYTCZUK, J.; KIERSTEAD, H. A.; ZHU, X. The map-coloring game. **The American Mathematical Monthly.** Taylor & Francis, v. 114, n. 9, p. 793–803, 2007.

BODLAENDER, H. L. On the complexity of some coloring games. **International Journal of Foundations of Computer Science.** World Scientific, v. 2, n. 2, p. 133–147, 1991.

BOHMAN, T.; FRIEZE, A.; SUDAKOV, B. The game chromatic number of random graphs. **Random Structures & Algorithms.** Wiley Online Library, v. 32, n. 2, p. 223–235, 2008.

BRAVO, R. S.; KLEIN, S.; NOGUEIRA, L. T.; PROTTI, F.; Sampaio, R. M. Partitioning extended P4-laden graphs into cliques and stable sets. **Information Processing Letters.** Elsevier, v. 112, n. 21, p. 829–834, 2012.

CHARPENTIER, C.; SOPENA, É. Incidence coloring game and arboricity of graphs. **International Workshop on Combinatorial Algorithms.** Springer, p. 106–114, 2013.

CHRISTEN, C. A.; SELKOW, S. M. Some perfect coloring properties of graphs. **Journal of Combinatorial Theory, Series B.** Academic Press, v. 27, n. 1, p. 49–59, 1979.

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

DINSKI, T.; ZHU, X. A bound for the game chromatic number of graphs. **Discrete Mathematics.** Elsevier, v. 196, n. 1, p. 109–115, 1999.

DUNN, C.; LARSEN, V.; LINDKE, K.; RETTER, T.; TOCI, D. The game chromatic number of trees and forests. **Discrete Mathematics and Theoretical Computer Science.** (S.I), v. 17, n. 2, p. 31–48, 2015.

ERDOS, P.; HAJNAL, A. On chromatic number of graphs and set-systems. **Acta Mathematica Academiae Scientiarum Hungarica.** Springer, v. 17, n. 1-2, p. 61–99, 1966.

FAIGLE, U.; KERN, W.; KIERSTEAD, H; TROTTER, W. On the game chromatic number of some classes of graphs. **Ars Combinatoria.** (S.I), v. 35, p. 143–150, 1993.

GARDNER, M. Mathematical Games. **Scientific American.** (S.I), v. 23, 1981.

GIAKOUMAKIS, V. P4-laden graphs: a new class of brittle graphs. **Information processing letters.** Elsevier, v. 60, n. 1, p. 29–36, 1996.

GRAPHS. Disponível em : <http://users.cecs.anu.edu.au/ bdm/data/graphs.html>. Acesso em: 29 Dez. 2018.

HAVET, F.; ZHU, X. The game Grundy number of graphs. **Journal of Combinatorial Optimization.** Springer, v. 25, n. 4, p. 752–765, 2013.

JAMISON, B.; OLARIU, S. A tree representation for P4-sparse graphs. **Discrete Applied Mathematics.** North-Holland, v. 35, n. 2, p. 115–129, 1992.

KIERSTEAD, H.; TROTTER, W. Planar graph coloring with an uncooperative partner. **Journal of Graph Theory.** Wiley Online Library, v. 18, n. 6, p. 569–584, 1994.

KRAWCZY, T.; WALCZAK, B. Asymmetric coloring games on incomparability graphs. **arXiv preprint arXiv:1503.04748.** (S.I), 2015.

SCHAEFER, T. J. On the complexity of some two-person perfect-information games. **Journal of Computer and System Sciences** (S.I), v. 16, n. 2, p. 185–225, 1978.

SIA, C. The game chromatic number of some families of Cartesian product graphs. **AKCE Int. J. Graphs Comb.** (S.I), v. 6, n. 2, p. 315–327, 2009.

SIDOROWICZ, E. The game chromatic number and the game colouring number of cactuses. **Information Processing Letters.** Elsevier, v. 102, n. 4, p. 147–151, 2007.

ZHU, X. The game coloring number of planar graphs. **Journal of Combinatorial Theory, Series B** Elsevier, v. 75, n. 2, p. 245–258, 1999.

ZHU, X. The game coloring number of pseudo partial k-trees. **Discrete Mathematics.** North-Holland, v. 215, n. 1-3, p. 245–262, 2000.

ZHU, X. Refined activation strategy for the marking game. **Journal of Combinatorial Theory, Series B.** Academic Press, v. 98, n. 1, p. 1–18, 2008.

APÊNDICE A - CÓDIGO PARA A VERIFICAÇÃO DA CONJECTURA DAS CORES PARA GRAFOS SIMPLES

Código utilizado para testar a conjetura de que se Alice ganha na versão clássica do jogo de coloração utilizando k cores, então ela também ganha utilizando $k' > k$ cores. A implementação foi feita em Python na versão 3.6.

```
import logging
import random

class Vertex:
        def __init__(self, number: int):
                self.number = number
                self.neighborhood = set()

        def __lt__(self, other):
                return self.number < other.number

        def __str__(self):
                return str(self.number)

        def __repr__(self):
                return str(self.number)


class Graph(object):
        def __init__(self):
                self.vertices = list()
                self.name = ""

        def create_from_string(self, text:str):
                text_to_graph = text.splitlines()
                self.name = text_to_graph.pop(0)
                text_to_graph = [x.strip() for x in text_to_graph]
                text_to_graph = [x.strip(";") for x in text_to_graph]
                incidence_list = [x.split(" : ") for x in text_to_graph]
                for x in incidence_list:
                        self.add_vertex(Vertex(int(x[0])))
                for x in incidence_list:
                        vertex_a = int(x.pop(0))
                        neighboors = x.pop().split(" ")
                        for vertex_b in neighboors:
                                if vertex_b:
                                        self.add_edge(vertex_a, int(vertex_b))

        def add_vertex(self, vertex):
                self.vertices.append(vertex)
                self.vertices.sort()

        def find_vertex(self, number: int) -> Vertex:
                for vertex in self.vertices:
                        if vertex.number == number:
                                return vertex
                return None

        def add_edge(self, vertex_a, vertex_b):
                if isinstance(vertex_a, int):
                        vertex_a = self.find_vertex(vertex_a)
                if isinstance(vertex_b, int):
                        vertex_b = self.find_vertex(vertex_b)
                if vertex_a in self.vertices and vertex_b in self.vertices:
                        vertex_a.neighborhood.add(vertex_b)
                        vertex_b.neighborhood.add(vertex_a)
                        return
                else:
                        raise Exception('Can not add edge from ' + str(vertex_a) + ' to ' +
                        str(vertex_b))

        def __str__(self):
                representation = 'Graph name: ' + self.name + '\n'
                representation += '\tList of vertices:\n'
                representation += '\t' + str(sorted(self.vertices)) + '\n'
                representation += '\tAdjacency List:\n'
                for vertex in sorted(self.vertices):
                        representation += '\t' + str(vertex) + ': ' +
                        str(sorted(vertex.neighborhood)) + '\n'
                return representation

        def max_degree(self) -> int:
                maximum = 0
                for vertex in self.vertices:
                        maximum = max(len(vertex.neighborhood), maximum)
```

```
                return maximum

class GraphEnumerator(object):
        def __init__(self, graph_file, starting_graph:int=0):
                self.current_graph = starting_graph
                self.graph_text = ""
                with open(graph_file, encoding="utf16") as file:
                        self.graph_text = file.read().split("\n\n")
                self.list_of_graphs = list()
                for text in self.graph_text:
                        graph = Graph()
                        graph.create_from_string(text)
                        self.list_of_graphs.append(graph)

        def __str__(self):
                representation = ""
                for item in self.list_of_graphs:
                        representation += str(item)
                return representation

class Coloring():
        def __init__(self, graph, number_colors):
                self.graph = graph
                self.color = {vertex : None for vertex in self.graph.vertices}
                self.max_colors = number_colors

        def copy(self, other):
                self.graph = other.graph
                self.color = dict(other.color)
                self.max_colors = other.max_colors

        def __str__(self):
                representation = 'Coloring:\n'
                representation += '\tNumber of colors available: ' +
                str(self.max_colors) + '\n'
                for vertex in sorted(self.graph.vertices):
                                representation += '\t' + str(vertex) + ': ' +
                                str(self.color[vertex]) + '\n'
                return representation.rstrip()

        def add_color(self, vertex, color: int):
                if isinstance(vertex, int):
                        vertex = self.graph.find_vertex(vertex)
                if self.color[vertex] is None and color < self.max_colors:
                        self.color[vertex] = color
                else:
                        raise Exception('Attempting to color ' + str(vertex) +
                        ' that has color ' + str(self.color[vertex]) +
                        ' with color ' + str(color))

        def remove_color(self, vertex):
                if self.color[vertex] is not None:
                        self.color[vertex] = None
                else:
                        raise Exception('Attempting to remove color ' + str(vertex) +
                        ' that has no color')

        def proper_coloring_test(self) -> bool:
                if self.__color_clash_list():
                        return False
                else:
                        return True

        def proper_color_list(self, vertex):
                if isinstance(vertex, int):
                        vertex = self.graph.find_vertex(vertex)
                if self.color[vertex] is None:
                        possible_colors = list()
                        for color in range(0, self.max_colors):
                                if color not in [self.color[neighbor] for neighbor in
                                vertex.neighborhood]:
                                        possible_colors.append(color)
                        return sorted(list(set(possible_colors)))
                else:
                        raise Exception('Testing if ' + str(vertex) + ' that has color ' +
                        str(self.color[vertex]) + ' can be proper colored.')

        def available_vertices(self):
                return [vertex for vertex in self.color.keys() if self.color[vertex] is None]

        def __color_clash_list(self) -> bool:
                clashes = list()
                for vertex in self.graph.vertices:
                        for neighbor in vertex.neighborhood:
                                if self.color[vertex] == self.color[neighbor]:
                                        clashes.append([vertex, neighbor])
                return sorted(clashes)

def game_coloring_grundy(graph: Graph, coloring: Coloring, player = 'Alice', turn = 0) -> bool:
        if not graph.vertices:
                return True
        if coloring.max_colors < 1:
```

```
                return False
        logging.debug('\nIn Turn (' + str(turn) + ') - Player is ' + player)
        logging.debug(coloring)
        if player == 'Alice':
                alice_wins = False
                available_vertices = coloring.available_vertices()
                if not available_vertices:
                        logging.debug('All vertices are colored.\n')
                        return True
                for vertex in available_vertices:
                        available_colors = coloring.proper_color_list(vertex)
                        if not available_colors:
                                logging.debug('\tImpossible to color vertex: ' + str(vertex))
                                return False
                        else:
                                new_coloring = Coloring(graph, coloring.max_colors)
                                new_coloring.copy(coloring)
                                new_coloring.add_color(vertex, available_colors[0])
                                logging.debug('\tIn Turn (' + str(turn) +
                                 ') Alice plays by coloring ' +
                                str(vertex) + ' with color ' + str(available_colors[0]))
                                alice_wins = game_coloring_grundy(graph, new_coloring, 'Bob',
                                turn + 1)
                                logging.debug('\nBack to Turn (' + str(turn) + ') -
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Player is - ' + player)
                                logging.debug(coloring)
                                if alice_wins:
                                        return True
                return False
        else:
                alice_wins = True
                available_vertices = coloring.available_vertices()
                if not available_vertices:
                        logging.debug('All vertices are colored.\n')
                        return True
                for vertex in available_vertices:
                        available_colors = coloring.proper_color_list(vertex)
                        if not available_colors:
                                logging.debug('\tImpossible to color vertex: ' + str(vertex))
                                return False
                        else:
                                new_coloring = Coloring(graph, coloring.max_colors)
                                new_coloring.copy(coloring)
                                new_coloring.add_color(vertex, available_colors[0])
                                logging.debug('\tIn Turn (' + str(turn) + ')
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Bob plays by coloring ' +
                                str(vertex) + ' with color ' + str(available_colors[0]))
                                alice_wins = game_coloring_grundy(graph,
                                new_coloring, 'Alice', turn + 1)
                                logging.debug('\nBack to Turn (' + str(turn) +
                                 ') - Player is - ' + player)
                                logging.debug(coloring)
                                if not alice_wins:
                                        return False
                return True
        return None

def game_grundy_number(graph: Graph) -> int:
        for i in range(1, graph.max_degree() + 2):
                coloring = Coloring(graph, i)
                if game_coloring_grundy(graph, coloring):
                        logging.info('\nFound The game grundy number: ' + str(i))
                        return i
                else:
                        logging.debug('\nImpossible to win (game grundy) with: ' + str(i) +
                         ' colors')
        return None

def game_coloring(graph: Graph, coloring: Coloring, player = 'Alice', turn = 0) -> bool:
        if not graph.vertices:
                return True
        if coloring.max_colors < 1:
                return False
        logging.debug('\nIn Turn (' + str(turn) + ') - Player is ' + player)
        logging.debug(coloring)
        if player == 'Alice':
                alice_wins = False
                available_vertices = coloring.available_vertices()
                if not available_vertices:
                        logging.debug('All vertices are colored.\n')
                        return True
                for vertex in available_vertices:
                        available_colors = coloring.proper_color_list(vertex)
                        if not available_colors:
                                logging.debug('\tImpossible to color vertex: ' + str(vertex))
                                return False
                        else:
                                logging.debug('\tList of available colors for vertex (' +
                                str(vertex) + '): ' + str(available_colors))
                                for color in available_colors:
                                        new_coloring = Coloring(graph, coloring.max_colors)
                                        new_coloring.copy(coloring)
```

```
                    new_coloring.add_color(vertex, color)
                    logging.debug('\tIn Turn (' + str(turn) +
                    ') Alice plays by coloring ' +
                    str(vertex) + ' with color ' + str(color))
                    alice_wins = game_coloring(graph, new_coloring,
                    'Bob', turn + 1)
                    logging.debug('\nBack to Turn (' + str(turn) + ') -
                                        Player is - ' + player)
                    logging.debug(coloring)
                    if alice_wins:
                        return True
            logging.debug('\tExhausted list of available colors for ('
            + str(vertex) + ')')
        return False
    else:
        alice_wins = True
        available_vertices = coloring.available_vertices()
        if not available_vertices:
            logging.debug('All vertices are colored.\n')
            return True
        for vertex in available_vertices:
            available_colors = coloring.proper_color_list(vertex)
            if not available_colors:
                logging.debug('\tImpossible to color vertex: ' + str(vertex))
                return False
            else:
                logging.debug('\tList of available colors for vertex (' +
                 str(vertex) + '): ' +
                str(available_colors))
                for color in available_colors:
                    new_coloring = Coloring(graph, coloring.max_colors)
                    new_coloring.copy(coloring)
                    new_coloring.add_color(vertex, color)
                    logging.debug('\tIn Turn (' + str(turn) +
                    ') Bob plays by coloring ' +
                    str(vertex) + ' with color ' + str(color))
                    alice_wins = game_coloring(graph, new_coloring,
                    'Alice', turn + 1)
                    logging.debug('\nBack to Turn (' + str(turn) + ') -
                                        Player is - ' + player)
                    logging.debug(coloring)
                    if not alice_wins:
                        return False
            logging.debug('\tExhausted list of available colors for ('
            + str(vertex) + ')')
        return True
    return None

def game_coloring_number(graph: Graph) -> int:
    for number in range(1, graph.max_degree() + 2):
        coloring = Coloring(graph, number)
        if game_coloring(graph, coloring):
            logging.debug('\nFound The game coloring number: ' + str(number))
            return number
        else:
            logging.debug('\nImpossible to win (game coloring) with: ' +
            str(number) + ' colors')
    return None

def game_coloring_conjecture(graph: Graph) -> bool:
    minimum_number = game_coloring_number(graph)
    logging.info('Testing conjecture (Alice + colors) with graph: ')
    logging.info(str(graph))
    logging.info('\tGraph game coloring number:' + str(minimum_number))
    logging.info('\tGraph maximum degree:' + str(graph.max_degree()))
    if minimum_number == graph.max_degree() + 1:
        logging.info('\tNo need to test (max degree == game coloring number)')
        return True
    for number_of_colors in range(minimum_number, graph.max_degree() + 2):
        coloring = Coloring(graph, number_of_colors)
        result = game_coloring(graph, coloring)
        logging.info('\tAlice has winning strategy with ' + str(number_of_colors) +
         ' colors: ' + str(result))
        if not game_coloring(graph, coloring):
            logging.info('\tThe conjecture (Alice + colors) is FALSE for this graph.')
            return False
    logging.info('\tThe conjecture (Alice + colors) is true for this graph.')
    return True

if __name__ == '__main__':
    logging.basicConfig(format='%(message)s', filename='strategyrun.log', filemode='w',
    level=logging.INFO)

    alice_colors_conjecture = True
    graph_enumerator = GraphEnumerator("Name Of File With Graph Instances")

    i = 0
    total = len(graph_enumerator.list_of_graphs)
    for graph in graph_enumerator.list_of_graphs:
        print("Testing graph " + str(i) + " of " + str(total))
        alice_colors_conjecture = game_coloring_conjecture(graph)
        if not alice_colors_conjecture:
```

```
                logging.critical("\n\nThe Conjecture 'Alice + colors' is FALSE!\n\n")
                break
        i += 1
graph_enumerator = GraphEnumerator("grafo9.txt")

i = 0
total = len(graph_enumerator.list_of_graphs)
for graph in graph_enumerator.list_of_graphs:
        print("Testing graph " + str(i) + " of " + str(total))
        alice_colors_conjecture = game_coloring_conjecture(graph)
        if not alice_colors_conjecture:
                logging.critical("\n\nThe Conjecture 'Alice + colors' is FALSE!\n\n")
                break
        i += 1
```