UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

BRUNO DA SILVA PINHO

OBTENDO O NÚMERO DE GRUNDY DE GRADES PARCIAIS

QUIXADÁ

2019

BRUNO DA SILVA PINHO

OBTENDO O NÚMERO DE GRUNDY DE GRADES PARCIAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Orientador: Prof. Me. Arthur Rodrigues Araruna

Coorientador: Prof. Me. Anderson Lemos da Silva

QUIXADÁ

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)

P723o Pinho, Bruno da Silva.
Obtendo o número de Grundy de grades parciais / Bruno da Silva Pinho. – 2019.
50 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2019.
Orientação: Prof. Me. Arthur Rodrigues Araruna.
Coorientação: Prof. Me. Anderson Lemos da Silva.

1. Teoria dos Grafos. 2. Coloração de grafos. 3. Algoritmos. I. Título.

CDD 005

BRUNO DA SILVA PINHO

OBTENDO O NÚMERO DE GRUNDY DE GRADES PARCIAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Aprovada em: __/__/____

BANCA EXAMINADORA

Prof. Me. Arthur Rodrigues Araruna (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Me. Anderson Lemos da Silva (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Wladimir Araújo Tavares
Universidade Federal do Ceará (UFC)

Prof. Dr. Paulo Henrique Macedo de Araújo
Universidade Federal do Ceará (UFC)

À minha mãe, por sua capacidade de sempre acreditar em mim.

AGRADECIMENTOS

Ao Prof. Me. Arthur Rodrigues Araruna por me orientar em meu trabalho de conclusão de curso.

Ao Prof. Me. Anderson Lemos da Silva por ter aceitado ser meu coorientador.

Aos meus pais e irmãos, por acreditarem sempre na minha capacidade e por nunca terem desistido de mim, me por terem me ajudado sempre.

Aos meus amigos Crislane, Marianna e Luís Henrique por estarem comigo deste de o começo.

E a turma minha turma de SI que foram os melhores colegas que já tive.

Agradeço a todos os professores por me proporcionar o conhecimento.

“A mente que se abre a uma nova ideia, jamais voltará ao seu tamanho original.”

(Albert Einstein)

RESUMO

O número de Grundy é a maior quantidades de cores que o algoritmo guloso de coloração consegue atribuir a um grafo. Para grafos em geral é difícil de se obter esse número. Então, neste trabalho, trabalhamos sobre uma classe de grafos denominada de grades parciais que, são subgrafos de grades, e descrevemos uma forma de obter o número de Grundy para essa classe. Neste trabalho também mostramos propriedades e estruturas que nos permitem construir uma solução algorítmica para esses tipos de grafos. Em específico, apresentamos duas soluções que podem ser utilizadas para encontrar o número de Grundy de grades parciais.

Palavras-chave: Número de Grundy. Coloração gulosa. Grades parciais.

ABSTRACT

The Grundy number is the larger amoust of colors that the greedy coloring algorithm can assign to a graph. For graphs in general, this number is difficult to obtain. In this work we presented a class of graphs called partial grids, which are subgraphs of grids, and a way to determine the Grundy Number for this class. We also show properties and structures that allow to build an algorithmic solution for these types of graphs. Specifically, we present two solutions taht can be used to determine the Grundy Number of partial grids.

Keywords: Grundy Number. Coloring. Partial Grids.

LISTA DE FIGURAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Grafos são estruturas abstratas bastante úteis na modelagem e representação de diversos tipos de problemas. Um problema bastante antigo e conhecido em Teoria dos Grafos é o problema de Coloração de Grafos (GAMA *et al.*, 2016).

A coloração de um grafo consiste em uma atribuição de cor aos seus vértices. Quando pede-se que vértices vizinhos tenham cores diferentes, chamamos essa atribuição de coloração própria. Esse problema se originou a partir do problema de coloração de mapas, que consistia em utilizar o menor número de cores para se colorir regiões de um mapa, tal que regiões que faziam fronteiras entre si possuíssem cores diferentes.

De modo geral, a coloração de grafos pode modelar problemas onde necessitamos que objetos conflitantes não estejam no mesmo grupo. Na modelagem, os vértices representam objetos, e as arestas representam os conflitos existentes entre os objetos (SILVA *et al.*, 2016). De acordo com Silva *et al.* (2016), diversos problemas podem ser modelados como problemas de coloração, tais como problemas de grade de horários de professores, alocação de tempo e frequência em sistemas de comunicação, alocação de registradores em um processador, controle de fluxo aéreo, dentre outros, possuindo assim um ramo bastante extenso de aplicações.

Uma forma de se obter uma coloração de um grafo é utilizando o Algoritmo Guloso de Coloração. Esse algoritmo considera que cada cor seja representada por um número natural, e considerando os vértices numa ordem dada como entrada, atribui a cada vértice nessa ordem, a menor cor que esteja disponível. A menor quantidade de cores que esse algoritmo atribui utilizando uma ordem ótima é igual ao número cromático. Do ponto de vista contrário, olhando para a maior quantidade de cores que esse algoritmo consegue atribuir a um grafo dada uma ordem de entrada que permite essa atribuição, temos o que é chamado de número de Grundy, também denominado de *número cromático guloso*. Uma coloração obtida por esse algoritmo onde se usa o máximo número de cores possível é chamada de coloração de Grundy.

De acordo com Effantin e Kheddouci (2007), existe uma aplicação para o número de Grundy na arquitetura de multiprocessadores. Por exemplo, suponha que há um conjunto de processos de tal forma que um processo P_i poderia ser computado se os processos $P_1, P_2, ..., P_{i-1}$ já estiverem computados. Tal regra em processos pode ser modelada por uma coloração de Grundy. Caso esses processos sejam computados de maneira distribuída, a solução para esse problema aplicada à topologia de comunicação entre os nós forneceria como resposta o processo mais complexo capaz de ser computado nessa topologia.

Para algumas classes de grafos, já se conhece o número de Grundy, como por exemplo os grafos bipartidos completos que, de acordo com Effantin e Kheddouci (2007), possuem o número de Grundy igual a 2. Para outras classes, conhecemos apenas limites para esse número. Em outros casos, é difícil de se determinar o número de Grundy. Um exemplo é mostrado em Sampaio (2012), onde é provado que encontrar o número de Grundy de um grafo cordal é um problema *NP*-difícil.

Uma classe de grafos bem conhecida para a qual sabemos determinar o número de Grundy facilmente é a de grades. Effantin e Kheddouci (2007) mostraram um resultado que nos permite derivar uma forma de obter esse número em tempo polinomial. Entretanto, os critérios utilizados para a análise sobre esses grafos não são suficientes para resolvermos o problema para seus subgrafos, que são chamados de grades parciais.

Isso posto, neste trabalho, vamos nos dedicar a estudar uma forma eficaz de se obter o número de Grundy para grades parciais.

O restante deste trabalho está organizado da seguinte maneira: O capítulo 2 apresenta notações e definições utilizadas nesse trabalho e o trabalho de Effantin e Kheddouci (2007). No capítulo 3 é apresentados alguns teoremas obtidos como resultados do problema em estudo. O capítulo 4 mostra os resultados computacionais obtidos com os teste das soluções algorítmicas propostas. No capítulo 5 são apresentadas as conclusões e trabalho futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, apresentaremos conceitos e definições de teoria dos grafos que são necessários para compreensão deste projeto. Esses conceitos são primordialmente advindos de Bondy *et al.* (1976) e West *et al.* (1996), que o leitor é encorajado a consultar caso necessite.

2.1 Grafos

Consideramos um grafo simples $G = (V, E)$, ao qual nos referimos neste texto apenas por *grafo*, como um par de conjuntos finitos denotados por $V(G)$ e $E(G)$, em que $E(G)$ é dado por pares não-ordenados $\{u, v\}$ de elementos distintos $u, v \in V(G)$.

Se $u \in V(G)$, o chamamos *vértice*. Se $\{u, v\} \in E(G)$, o chamamos de *aresta*. Também nos referimos a $\{u, v\}$ apenas por uv. Além disso, se uv é uma aresta, dizemos que os vértices u e v são suas *extremidades* e que eles são *adjacentes* entre si. Se um grafo possui apenas um único vértice o chamamos de grafo *trivial*. A *vizinhança* de um vértice v, denotada por $N(v)$, é o conjunto de todos os vértices adjacentes a v.

O *grau de um vértice* v, $d_G(v)$, é o número de arestas incidentes a v em G. Denotamos por $\delta(G)$ e $\Delta(G)$ o menor e o maior valor de grau entre os vértices de G, respectivamente.

Dizemos que um grafo H é um *subgrafo* de G, denotado por $H \subseteq G$, se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$.

A *ordem* de um grafo G, denotada por $n(G)$, é o seu número de vértices, e seu *tamanho* denotado por $e(G)$, é o seu número de arestas.

Um *caminho* P é um grafo tal que seus vértices podem ser ordenados de forma que dois vértices são adjacentes se, e somente se, eles são consecutivos nessa ordem. Sabemos que $|E(P)| = |V(P)| - 1$ (WEST *et al.*, 1996). Dizemos que P_n é um caminho com n vértices.

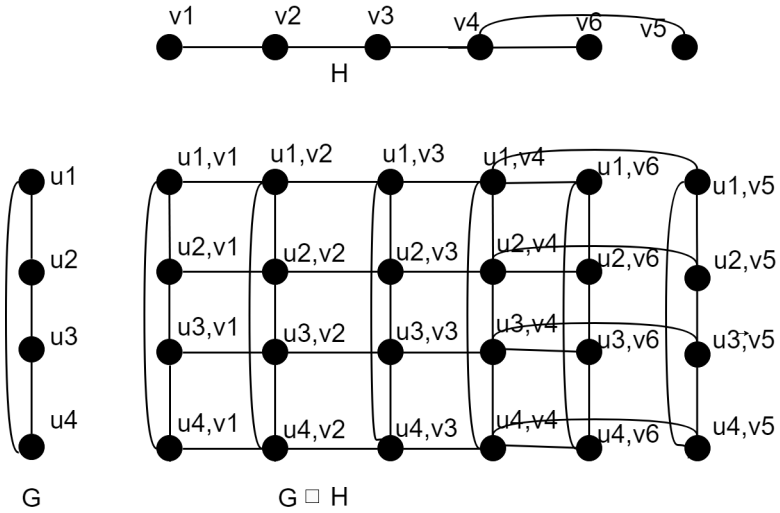
Um *ciclo* C é um grafo em que o número de vértices e de arestas é o mesmo e cujos vértices podem ser dispostos em ordem circular de forma que dois vértices são adjacentes se, e somente se, são consecutivos nessa ordem.

Um grafo G é *bipartido* se ele for trivial ou se seu conjunto de vértices pode ser particionado em dois subconjuntos disjuntos X e Y tais que toda aresta de G tem uma extremidade em X e outra em Y. Dessa forma, X e Y são *conjuntos independentes*, nome dado para um conjunto de vértices que não têm arestas entre si. Sabemos que um grafo é bipartido, se e somente se, não possui ciclos de tamanho ímpar (WEST *et al.*, 1996).

2.2 Produto cartesiano de grafos

Sejam os grafos $G = (V_1, E_1)$ e $H = (V_2, E_2)$ tais que $\{u_1, ..., u_n\}$ são os vértices de G e $\{v_1, ..., v_n\}$ são os vértices de H. O *produto cartesiano* entre G e H, denotado por $G \square H$, é o grafo com o conjunto de vértices $V = V_1 \times V_2$, com (u_i, v_j) adjacente a (u_l, v_p) sempre que u_i é adjacente a u_l em G e $v_j = v_p$ em H ou $u_i = u_l$ em G e v_j é adjacente a v_p em H (SOUZA, 2016). A Figura 1 mostra o produto cartesiano entre dois grafos G e H.

Figura 1 – Produto cartesiano $G \square H$

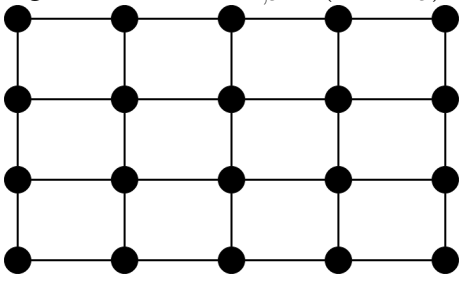


Fonte: Elaborada pelo Autor

2.2.1 Grade

Uma *grade* é um grafo $G_{m,n}$ definido como o produto cartesiano de dois caminhos P_m e P_n, de forma que $G_{m,n} = (P_m \square P_n)$. A Figura 2 representa um exemplo de uma grade formada por dois caminhos P_4 e P_5.

Figura 2 – Grade $G_{4,5} = (P_4 \square P_5)$

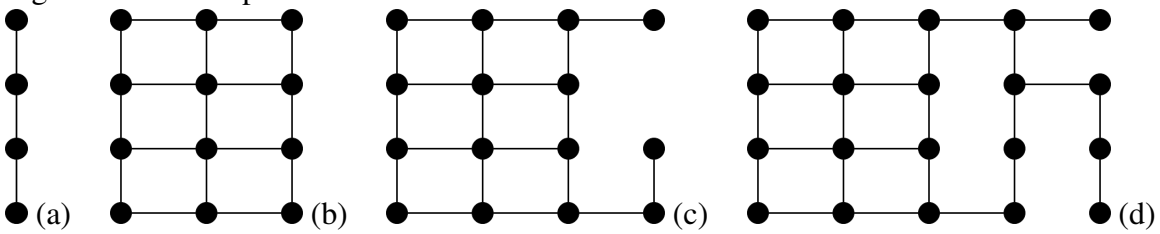


Fonte: Elaborada pelo Autor

2.2.2 Grade Parcial

Um grafo H é uma *grade parcial* se ela é um subgrafo de uma grade $G_{n,m}$. Na Figura 3 são mostradas quatro grades parciais que são subgrafos do grafo da Figura 2.

Figura 3 – Grades parciais



Fonte: Elaborada pelo Autor

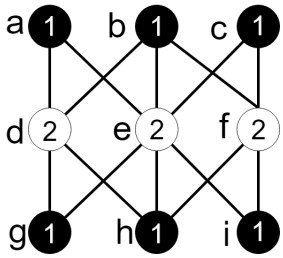
2.3 Coloração

Dado um grafo $G = (V, E)$, uma *coloração* de G, é uma função $c : v \to \mathbb{N}$ que associa a cada vértice do grafo um número natural, denominado cor. Se uma coloração do grafo G usa exatamente k cores, podemos chamá-la de k-coloração de G (OLIVEIRA, 2011). Uma k-coloração pode ser vista como uma partição $S = \{S_1, ..., S_k\}$ de $V(G)$ em k conjuntos disjuntos onde cada conjunto S_i contém os vértices coloridos com a cor i, para todo $i \in \{1, ..., k\}$. Os conjuntos S_i são as *classes de cores* dessa coloração (OLIVEIRA, 2011). Uma coloração c é *própria* se a nenhum par de vértices adjacentes é atribuída a mesma cor.

2.3.1 Número Cromático

O menor inteiro k para o qual um grafo G possui uma k-coloração própria é o *número cromático* de G, denotado por $\chi(G)$. A Figura 4 apresenta um grafo colorido com duas cores. Note que não podemos colorir o grafo com menos cores e ainda termos uma coloração própria. Devido ser a menor quantidade de cores que é possível atribuir a esse grafo de forma que ele possua uma coloração própria, dizemos então que $\chi(G) = 2$.

Figura 4 – $\chi(G) = 2$



Fonte: Elaborada pelo Autor

2.3.2 Algoritmo Guloso de Coloração

Algumas das principais abordagens algorítmicas utilizadas para o problema de coloração são baseadas em métodos gulosos que realizam a coloração sequencial dos vértices usando uma função gulosa (SILVA *et al.*, 2016). Uma dessas abordagens é o *Algoritmo Guloso de Coloração*, também conhecido como *Algoritmo Sequencial*, que é descrito no Algoritmo 1.

Algoritmo 1: ALGORITMO GULOSO DE COLORAÇÃO

Entrada: Grafo $G = (V,E)$ e ordem $\theta = v_1, v_2, ..., v_n$ de $V(G)$

Saída: Coloração própria c de G

```
1 início
2     para cada v_i ∈ θ faça
3         c(v_i) ← a menor cor não utilizada em N(v_i) ∩ {v_1, ..., v_{i-1}}
4     fim
5     retorna c
6 fim
```

Fonte: Baseado no trabalho de Oliveira (2011)

A esse algoritmo é dado como entrada um grafo G e uma ordem θ dos vértices de G. Em cada iteração ele seleciona um vértice seguindo a sequência de θ. Para esse vértice é atribuida a menor cor que não foi atribuída aos seus vizinhos já coloridos. De acordo com Oliveira (2011), o Algoritmo Guloso de Coloração gera sempre colorações próprias para um grafo.

Dizemos que uma coloração c é gulosa se ela pode ser gerada pelo Algoritmo 1. Se a coloração gerada por tal algoritmo possui k cores, também podemos chamá-la de *k-coloração gulosa* (OLIVEIRA, 2011). Podemos notar que tal algoritmo pode apresentar

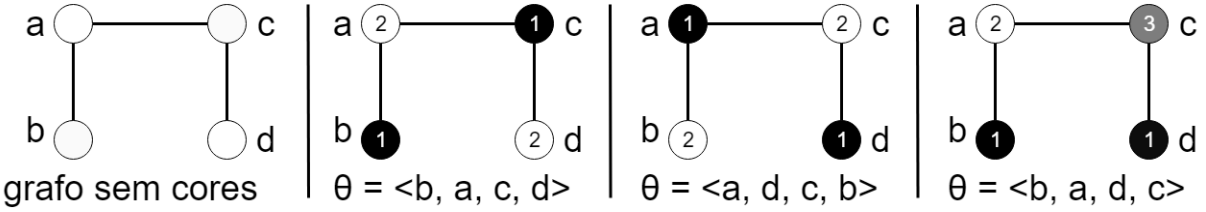
diferentes colorações para um mesmo grafo, a depender apenas da ordem informada para os vértices.

Segundo Neto e Gomes (2015), o número mínimo de cores que o Algoritmo Guloso consegue atribuir a um grafo é o número cromático. Sendo assim, existe uma ordem de vértices que faz com que o Algoritmo 1 atribua uma coloração ao grafo com a quantidade de cores igual ao número cromático.

O número cromático parametriza o comportamento do melhor caso da coloração gerada pelo o Algoritmo 1. Dessa forma, o número de cores de qualquer coloração gerada por tal algoritmo é um limite superior pra o número cromático.

Na Figura 5 são mostradas três diferentes colorações gulosas para o mesmo grafo. Nessa figura podemos perceber o impacto da ordem considerada na coloração obtida.

Figura 5 – Coloração gulosa obtida com diferentes ordens θ de vértices



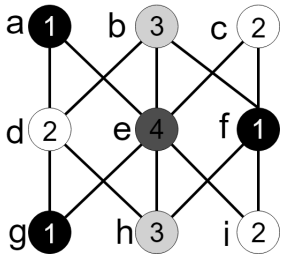
Fonte: Baseado no trabalho de Silva *et al.* (2016)

2.3.3 Número de Grundy

O *número de Grundy* de um grafo G é o maior k tal que G tem uma k-coloração gulosa. O problema de Coloração Gulosa consiste em determinar, dentre todas as possíveis ordenações de $V(G)$, a maior quantidade de cores que o Algoritmo 1 utiliza (OLIVEIRA, 2011). Denominamos essa quantidade de número guloso de G ou número de Grundy, que é denotado por $\Gamma(G)$. O número de Grundy e sua relação com o número cromático parametriza o pior caso da coloração que Algoritmo Guloso de Coloração consegue em um grafo. Dessa forma, temos que $\chi(G) \leq \Gamma(G)$, determinando o quão ruim é esse limite superior para o número cromático.

De acordo com Sampaio (2012), um limite superior para o número de Grundy de qualquer grafo é $\Gamma(G) \leq \Delta(G) + 1$. Na Figura 6 é mostrada uma coloração gulosa. Note que foi possível colorir o grafo com quatro cores, e essa é a ordem cuja a quantidades de cores é a maior que o Algoritmo 1 consegue utilizar para colorir o grafo da Figura 6. Então podemos concluir que $\Gamma(G) = 4$.

Figura 6 – $\Gamma(G) = 4$



Fonte: Elaborada pelo Autor

2.4 Número de Grundy e Grades

Como mostrado por Sampaio (2012), um limite para o número de Grundy em grafos $\Gamma(G) \leq \Delta(G) + 1$. Effantin e Kheddouci (2007), mostram que uma grade G possui $\Delta(G) = 4$, então o limite superior para tal grafo é $\Gamma(G) \leq 5$. Effantin e Kheddouci (2007), para uma grade $G_{n,m}$, se $n = 2$ ou se n e m forem iguais 3, temos que o número de Grundy é no máximo 4; caso contrário temos que é sempre 5.

Como dito anteriormente, toda grade parcial é subgrafo de uma grade e, por definição, uma grade é subgrafo de si própria, logo temos que toda grade também é uma grade parcial. Seja H uma grade parcial que é subgrafo de uma grade G. De acordo com Ferreira (2003), toda grade é uma grafo bipartido, logo, G é bipartido. Para gerar H a partir de G, apenas foram apagados vértices e/ou arestas. Remover vértices e/ou arestas não pode gerar novos ciclos, pelo contrário, só pode desfazer ciclos que já existem. Assim, todo ciclo em H também é um ciclo de G. Um grafo é bipartido se e somente não contém ciclo de tamanho ímpar (WEST *et al.*, 1996), logo, todos os ciclos de G e consequentemente de H têm tamanho par. Como todos os ciclos de H têm tamanho par, H também é bipartido. Desse modo, além de toda grade ser uma grade parcial, toda grade parcial é bipartida. Sendo assim, sejam $\mathscr{G}$, $\partial\mathscr{G}$ e $\mathscr{B}$ as classes de grafos grades, grades parciais e bipartidos, respectivamente, temos que $\mathscr{G} \subseteq \partial\mathscr{G} \subseteq \mathscr{B}$.

Como demonstrado em Sampaio (2012), encontrar o número de Grundy para grafos bipartidos é NP-difícil. Porém, como mostrado em Effantin e Kheddouci (2007), determinar o número de Grundy para grades é fácil, pois pode ser feito em tempo polinomial. Assim sendo, é interessante o estudo do problema para a classe das grades parciais, pois não foram encontrados muitos resultados sobre o mesmo na literatura. Além disso, por a classe das grades parciais ser uma superclasse das grades e subclasse dos bipartidos, faz sentido o estudo da dificuldade computacional do problema para esse tipo de grafo.

Também podemos frisar de que, pelo o fato de uma grade parcial H ser um subgrafo

de uma grade G, temos que o número de vértices e arestas de H pode ser menor do que o de G, já que para termos um subgrafo podemos manter ou remover arestas e/ou vértices do grafo original. Pelo fato de podermos remover arestas e vértices, estamos diminuindo a quantidade de adjacências dos vértices, o que faz com que a grade parcial possa ter uma quantidade de cores menor do que a da grade original. Assim, temos o seguinte limite: $\Gamma(H) \leq \Gamma(G)$.

2.5 Número de Grundy de grafos

Em Effantin e Kheddouci (2007) são mostrados alguns limites para o número de Grundy em algumas classes de grafos e produtos cartesianos de grafos. Ele também mostra que, para caminhos, ciclos e bipartidos completos, o número de Grundy é fixo. Em particular é determinado o valor exato do número de Grundy para malhas com n dimensões e algumas malhas toroidais com n dimensões.

No trabalho é mostrado que para o produto cartesiano entre dois caminhos $P_n \square P_m$, definido anteriormente como sendo uma grade $G_{n,m}$, o número de Grundy de G é menor ou igual a 4 se $n = 2$ ou se $n = m = 3$, caso contrário, $\Gamma(G) = 5$.

Nesse mesmo trabalho, Effantin e Kheddouci (2007) apresentam um algoritmo recursivo que, dado um número k, consegue construir com o menor número de arestas todos os grafos que possuem número de Grundy igual a k. A ideia principal do algoritmo desse trabalho é começar com uma árvore com 2^{k-1} vértices e então unir alguns vértices que têm a mesma cor. Pela computação de todos os agrupamentos possíveis, é encontrado um conjunto de grafos com o número de Grundy igual a k. Esse processo é repetido recursivamente tanto quanto possível para cada grafo nesse conjunto.

3 RESULTADOS

Nesta capitulo, apresentamos os resultados desenvolvidos neste trabalho que culminam em duas estratégias algorítmicas de determinação do número de Grundy para uma grade parcial.

Ambas as estratégias se baseiam em identificar estruturas dentro do grafo informado, que chamamos de *gadget*, cuja existência compõe condição necessária e suficiente para a existência de uma coloração gulosa com um certo número de cores. Em grades parciais, dado que o grau máximo é limitado pelo grau máximo de uma grade, só são possíveis colorações gulosas que geram entre uma e cinco cores. Dessa forma, definimos *gadgets* para cada valor de coloração.

Consideramos, nesse caso, a informação de que estamos buscando essas estruturas em grades parciais, e fazemos uso desse fato nas demonstrações de necessidade e suficiência.

3.1 Definições

A seguir definimos os *gadgets* para 3, 4 e 5 cores para depois demonstrarmos que eles identificam colorações gulosas com as respectivas cores. As demonstrações se baseiam em demonstrar que, se existirem tais estruturas dentro do grafo em questão de forma que podemos colori-las de forma gulosa com a quantidade de cores desejada, então o grafo como um todo admite uma tal coloração gulosa e, de forma recíproca, uma coloração gulosa do grafo exibirá tal estrutura. Os *gadgets* para 1 e 2 cores são triviais e são discutidos mais à frente.

Dessa forma, considere um grade parcial G.

Definição 3.1.1 *Um* gadget *de 3 cores para G é um P_4 induzido.*

Definição 3.1.2 *Um* gadget *de 4 cores para G é um grafo H formado por um $P_6 = \langle a,b,c,d,e,f \rangle$ onde os dois vértices centrais c e d tenham grau pelo menos 3.*

Esse H possui ainda outros dois vértices, c_1 vizinho a c, e d_1 vizinho a d, que não fazem parte do P_6, e de forma que c_1 não é vizinho a d_1 em G.

Definição 3.1.3 *A* vizinhança *de um conjunto de vértices S, denotada por $N(S)$, é dada por* $N(S) = \bigcup_{v \in S} N(v)$.

Definição 3.1.4 *A* primeira vizinhança *de um vértice v, denotada por $N^1(v)$, é dada por* $N^1(v) = N(v)$.

Definição 3.1.5 *A* segunda vizinhança *de um vértice v, denotada por* $N^2(v)$*, é dada por* $N^2(v) = N(N^1(v))$.

Definição 3.1.6 *Chamamos de um* backbone *um* P_4 *induzido onde os dois vértices das pontas têm pelo menos grau 3 em G, e os dois vértices centrais têm grau 4 em G. Além disso, para cada vértice nesse* P_4*, ao menos um de seus vizinhos que não esteja no* P_4 *tem que ter grau pelo menos 2.*

Definição 3.1.7 *Um* gadget *de 5 cores para G é um grafo H induzido por um* backbone *e por suas primeira e segunda vizinhanças. Dessa forma,* $V_H = \{N^1(V(backbone)) \cup N^2(V(backone)) \cup V(backbone)\}$ *e* $H = G[V_H]$.

3.2 Teoremas

Os teoremas apresentados nesta seção são validos considerando G como uma grade parcial.

3.2.1 Gadget de 3 cores

Teorema 3.2.1 *Existe uma coloração gulosa de G com pelo menos 3 cores se, e somente se, em G existe um* gadget *de 3 cores.*

$(\Rightarrow)$

Se G possui uma coloração gulosa com pelo menos 3 cores, então G deve possuir ao menos um vértice com a cor 3, esse vértice por sua vez tem que ser vizinho de pelo menos um vértice que possua a cor 1 e outro que possua a cor 2. O vértice que tem a cor 2 deve ser vizinho de pelo menos um outro vértice com cor 1.

Sejam w, x e y os vértices com cor 1, 3 e 2. O vértice w não pode ser vizinho de y já que isso formaria um ciclo ímpar. Logo y, que possui a cor 2, tem que ter um outro vértice como vizinho com cor 1. Seja z esse novo vértice. Se z fosse vizinho de w, teríamos um ciclo ímpar, logo eles não podem ser vizinhos. Sendo assim, existe em G um caminho induzido com quatro vértices $P_4 = \langle w,x,y,z \rangle$, logo temos um *gadget* de 3 cores.

($\Leftarrow$)

Como definido, o *gadget* de 3 cores é um caminho induzido com 4 vértices. Sejam esses vértices w,x,y,z, em que w é vizinho de x, x é vizinho de y e y é vizinho de z. Queremos mostrar que este P_4 possui uma coloração gulosa com pelo menos 3 cores. Dessa forma, basta dar uma ordem para os vértices do P_4 de modo que o Algoritmo Guloso de Coloração consiga colorir com 3 cores esse caminho. Sendo assim se dermos a ordem $\{w,z,y,x\}$ o 1 irá colorir w com cor 1, z com cor 1, já que esses dois vértices não possuem nenhum vizinho com a cor 1 e essa é a menor cor que se pode atribuir a esses vértices, y será colorido com a cor 2, já que essa é a menor cor disponível para esse vértice pois é vizinho de um vértice com cor 1 e por fim irá colorir x com a cor 3 pois essa é a menor cor disponível para esse vértice, já que ele é vizinho de um vértice com cor 1 e outro com cor 2. Dessa forma conseguimos colorir o *gadget* com 3 cores. Com isso conseguimos estender a ordem dada de forma a colorir o grafo completo, bastando para os vértices de G, colorir primeiro os vértices de G equivalentes aos do *gadget* na ordem dada e seguir colorindo os restantes dos vértices em qualquer ordem. Como os primeiros vértices de G a ser coloridos são os mesmos que geram a 3-coloração gulosa no *gadget*, temos que G será colorido gulosamente com pelo menos 3 cores.

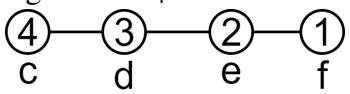
3.2.2 Gadget de 4 cores

Teorema 3.2.2 *Existe uma coloração gulosa de G com pelo menos 4 cores se, e somente se, em G existe um* gadget *de 4 cores.*

($\Rightarrow$)

Para termos uma coloração gulosa com pelo menos 4 cores, deve existir ao menos um vértice com a cor 4. Seja c esse vértice. Para que ele tenha a cor 4, c deve ter pelo menos três outros vértices vizinhos, que possuem a cor 3, 2 e 1. Seja d o vértices com cor 3 que é vizinho a c. Como d tem a cor 3, ele deve ser vizinho de pelo menos dois outros vértices, um com cor 1 e outro com cor 2. Seja e o vértice com cor 2; ele deve ter pelo menos um vizinho com cor 1. Seja f o vértice com cor 1 que é vizinho a e. Como pode ser visto na Figura 7,

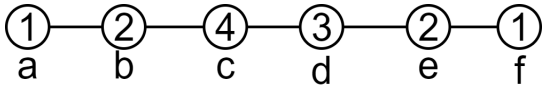
Figura 7 – P_4



Fonte: Elaborada pelo Autor

o vértice e não pode ser vizinho de c pois formaria um ciclo ímpar. Logo c tem um vizinho com cor 2 que é diferente de e. Seja b o vértice com a cor 2 como já dito, um vértice com cor 2 deve ter pelo menos um vértice vizinho com cor 1. Seja a esse vértice com cor 1 que é vizinho a b.

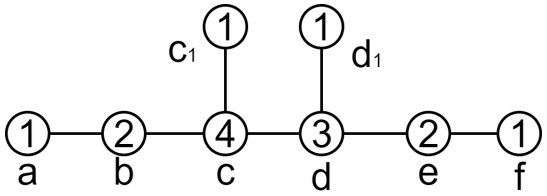
Figura 8 – P_6



Fonte: Elaborada pelo Autor

O vértice a não pode ser vizinho de c, e o vértice f não pode ser vizinho de d, pois formariam um ciclo ímpar. Logo c possui um vizinho que com a cor 1, e o vértice d também possui um vizinho com cor 1. Seja c_1 o vértice com cor 1 vizinho a c, e seja d_1 o vértice com cor 1 vizinho a d, d_1 não pode ser vizinho a c_1 já que eles devem ter a cor 1. Dessa forma, temos que o vértice d_1 pode ser ou não igual ao vértice a, e o vértice c_1 pode ser ou não igual ao vértice f. Sendo assim evidenciamos um *gadget* de 4 cores.

Figura 9 – *gadget* de 4 cores



Fonte: Elaborada pelo Autor

($\Leftarrow$)

Para mostrarmos que existe uma 4-coloração gulosa em G, se existe um *gadget* de 4 cores , bastar dar uma ordem para os vértices de forma que o algoritmo de coloração gulosa consiga colorir esse *gadget* usando 4 cores.

Como dito, o *gadget* de 4 cores possui um P_6. Sejam a, b, c, d, e, f os vértices que formam o P_6 de forma ordenada. Sejam c_1 o vértice que é vizinho a c e seja d_1 o vértice que é

vizinho a d, de forma que c_1 e d_1 não sejam vizinhos. Dessa forma podemos criar uma sequência com esses vértices de forma que o algoritmo de coloração gulosa consiga dar 4 cores para o P_6. Assim criamos a seguinte ordem de vértices $\{a, f, c_1, d_1, b, e, c, d\}$.

Caso o vértice c_1 seja igual a f ou o vértice d_1 seja igual a a, basta não colocá-los na ordem pois já estão considerados. Dessa forma o algoritmo guloso 1 irá colorir os vértices a, f, c_1, d_1, com a cor 1, já que eles não são vizinhos entre si, e não possuem vizinhos já coloridos; os vértices b e e serão coloridos com a cor 2, já que possuem cada um um vizinho que já foi colorido com 1; e por fim chegamos aos vértices c e d. Como c tem apenas vizinhos já coloridos com a cor 2 e 1, ele receberá a cor 3, restando para d ser atribuída a cor 4, já que possui um vizinho já colorido com cada cor 1, 2 e 3.

Assim sendo, o Algoritmo Guloso de Coloração irá colorir esse *gadget* com 4 cores. Com isso, conseguimos estender a ordem dada de forma a colorir o grafo completo, bastando para os vértices de G, colorir primeiro os vértices de G equivalentes aos do *gadget* na ordem dada e seguir colorindo os restantes dos vértices em qualquer ordem. Como os primeiros vértices de G a ser coloridos são os mesmos que geram a 4-coloração gulosa no *gadget*, temos que G será colorido de maneira gulosa com pelo menos 4 cores.

3.2.3 Gadget de 5 cores

Teorema 3.2.3 *Existe uma coloração gulosa de G com 5 cores se, e somente se, em G existe um* gadget *de 5 cores tal que seja possível atribuir* de forma própria *as cores 1 e 2 a alguns vértices fora do* backbone *de forma que:*

1. *Cada vértice do backbone possua ao menos um vizinho de cor 1 e um de cor 2.*
2. *Cada vértice colorido com cor 2 possua ao menos um vizinho de cor 1.*
3. *Os demais vértices não precisam ser coloridos.*

$(\Rightarrow)$

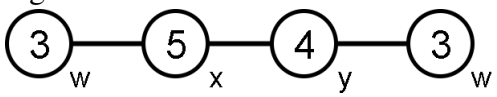
Se G possui uma coloração gulosa com pelo menos 5 cores, então existe um vértice que possui a cor 5. Seja x esse vértice. Esse x deve ser vizinho de pelo menos 4 vértices, com as cores 1, 2, 3 e 4. Seja y o vértice que possui a cor 4. Por sua vez y deve ser vizinho de pelo menos três vértices com cores 1, 2 e 3. Seja z o vértice com cor 3 vizinho a y. Nomearemos por w o vértice com a cor 3 vizinho a x.

Note que w e z não podem ser o mesmo vértice pois G conteria um ciclo impar

$\langle y, z, w \rangle$, e isso não ocorre em grades parciais.

Dessa forma, montamos um P_4 com os vértices w, x, y, z. Esse P_4 será o *backbone* do gadget que estamos construindo. Vamos dizer que ele pertence a um conjunto B. A Figura 10 ilustra o *backbone*.

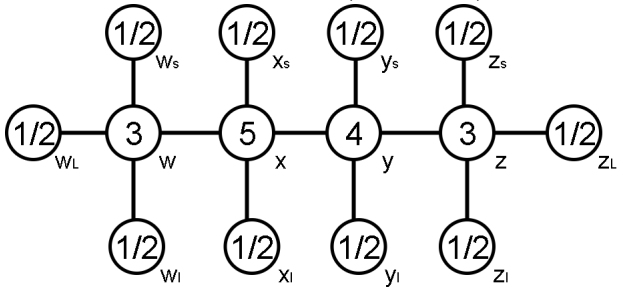
Figura 10 – *backbone*



Fonte: Elaborada pelo Autor

Agora, temos que cada vértice pertencente a B, devem ter dois vizinhos que não estão em B, exceto os vértices w e z que podem possuir um vizinho a mais. Para cada vértice em B vamos ter pelo menos um vizinho com a cor 1 e pelo menos um vizinho com a cor 2, como mostra a Figura 11. Note que na figura temos que os vértice que pertencem a $N(B)$ precisam necessariamente ter a cor 1 ou 2, dadas as cores atribuídas aos vértices de B. Sempre que um vértice em B tiver um vizinho com a cor 1, obrigatoriamente ele vai ter um outro vizinho com a cor 2. Seja a primeira vizinhança de B como sendo $N^1(B) = \{x_S, x_I, y_S, y_I, z_S, z_L, z_I, w_S, w_L, w_I\} - V(B)$ como mostra a Figura 11.

Figura 11 – $backbone \cup N^1(backbone)$

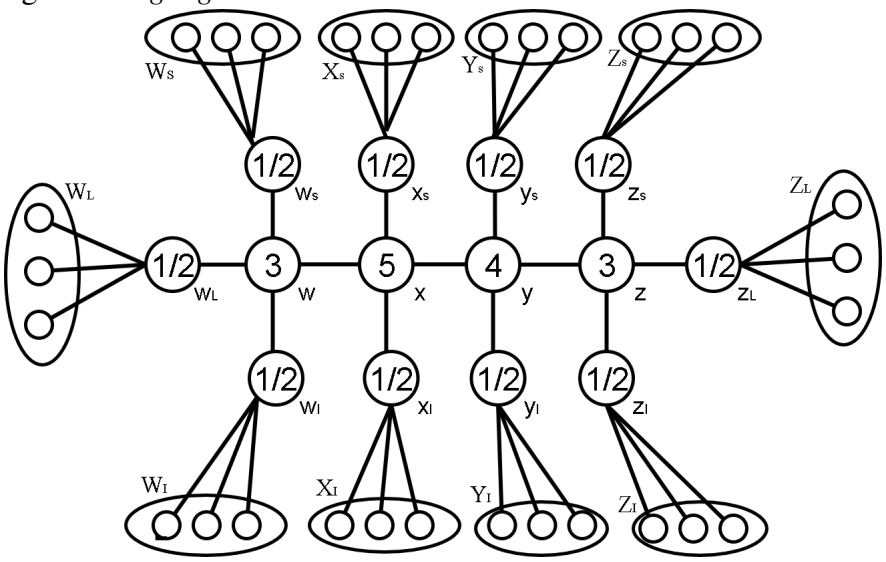


Fonte: Elaborada pelo Autor

Observe que os vértices escolhidos para serem coloridos com a cor 2, devem ter um vizinho com a cor 1 já que a coloração obtida é gulosa. Porém, os vértices com cor 2 e cor 1 que são vizinhos entre si, não podem ter o mesmo vizinho em B pois isso formaria um C_3.

Como sabemos, em uma grade parcial cada vértice pode ter no máximo quatro vizinhos. Sendo assim, $\forall V(G) \in N^1(B)$ que tem a cor 2 tem no máximo 3 vizinhos que não esta em B, e deve existir pelo menos um desses vizinho que podemos da a cor 1.

Figura 12 – *gadget* de 5 cores



Fonte: Elaborada pelo Autor

Vamos chamar de $N^2(B)$ como sendo o conjunto de vértices tal que $N^2(B) = \{N(N^1(B))\}$. Note que alguns ou todos os vértices de $N^1(B)$ podem está em $N^2(B)$ deste que exista alguma aresta entre os vértices de $N^1(B)$. Por exemplo, pode haver uma aresta de w_S para x_S, se isso ocorre então um deles tem a cor 1, e o outro a cor 2, logo $x_S \subset N(w_S)$ e $w_s \subset N(x_S)$, sendo assim $\{x_S, w_S\} \subset N^2(B)$. Caso algum vértice em $N^1(B)$ possua apenas vizinhos em B esse vértice obrigatoriamente só poderá ter a cor 1. Dessa forma, temos um *gadget* de 5 cores que respeita as restrições de coloração dadas.

($\Leftarrow$)

Se for possível atribuir cores 1 e 2 de forma própria para alguns vértices fora do *backbone* de forma que todo vértice do *backbone* tenham exatamente um vizinho com a cor 2 e um com a cor 1, e todos os vértices que tenham a cor 2 tenha pelo menos um vizinho na cor 1. Dessa forma, se tentarmos atribuir cores para os dois vértice que estão na extremidades do *backbone* só poderemos atribuir a cor 3 a eles já que esses vértices só possuem vizinhos com as cores 1 e 2. E, em seguida, tentarmos atribuir cores para os dois vértices centrais do *backbone*, um desses vértices terá a cor 4 já que ele tem vizinhos nas cores 1, 2 e 3, e o outro terá a cor 5 já que ele vai ter vizinhos das cores 1, 2, 3 e 5. Dessa forma, temos que *gadget* de 5 cores possui um coloração gulosa com 5 cores. Logo conseguimos dar uma ordem para os vértice de G de forma que nessa ordem venha primeiro os vértices do *gadget* que possuem cor 1, em seguida os que possuem cor 2, depois os que têm cor 3, e por fim adicionamos o restante dos

vértices do grafo nessa ordem, temos que conseguimos colorir G com pelo menos 5 cores de forma gulosa, já que os vértices iniciais dessa ordem são os mesmo que geraram a 5 coloração gulosa no *gadget*.

4 RESULTADOS COMPUTACIONAIS

4.1 Algoritmo

Em nosso trabalho não nos preocupamos em verificar se o grafo possui uma coloração gulosa com cores 1 e 2 , pois todo grafo G não-nulo tem $\Gamma(G) \geq 1$, e já para ter $\Gamma(G) \geq 2$ é suficiente que exista pelo menos uma aresta. Essas condições são triviais de serem testadas.

Com os algoritmos **Estratégia 1**, descrito na subseção 4.1.4, e **Estratégia 2**, descrito na subseção 4.1.5, apresentados neste trabalho, iremos fazer uma busca pelo grafo tentando encontrar os *gadgets* de 3, 4 e 5 cores. Encontrar esses *gadgets* significa dizer que o grafo possui $\Gamma(G)$ maiores ou iguais a 3, 4 e 5 respectivamente.

Uma vez possuindo algum *gadget* podemos facilmente construir uma ordem para fornecer uma coloração com o mínimo de cores correspondente. Basta apenas montá-la de forma que os vértices do *gadget*, que são os mesmos do grafo, sejam colocados na ordem onde os vértices com cores menores sejam listados primeiro do que os que possuem uma cor maior do que eles, e por fim, adicionamos os vértices que não estão coloridos segundo o *gadget*. Em seguida, basta passarmos essa ordem e o grafo para o Algoritmo Guloso de Coloração, que ele ira colorir todo o grafo.

Nos algoritmos abaixo, os grafos têm seus vértices enumerados de 0 a $n-1$ onde n é a quantidade de vértices desse grafo. Se temos um caminho com n vértices P_n, usamos $P_n[1]$, para nos referirmos ao primeiro vértice desse caminho, $P_n[2]$ para o segundo, e assim por diante.

4.1.1 Encontrar gadget *de 3 cores*

O algoritmo 2 é usado para procura um *gadget* de 3 cores $G3$ em uma grade parcial G. Neste algoritmo, temos que para cada vértice v_i de $V(G)$ tentamos montar um P_4 induzido. O processo mostrado na linha 3 do algoritmo 2, é um processo enumerativo em que dentre os vizinhos de um vértice w é escolhido um vértice x; dentre os vértices em $N(x) - \{w\}$, é escolhido um vértice y; e dentre $N(y) - \{x, w\}$ é escolhido um vértice z, dessa forma obtendo um P_4 induzido.

Para que não haja repetição de caminhos no processo de encontrar um P_4, é verificado se o primeiro vértice é menor que o último vértice do caminho numa ordem total fixa sobre os vértices do grafo. Por fim, pintamos os vértices do P_4 como mostra o algoritmo 2 e retornamos o *gadget* de 3 cores $G3$.

Algoritmo 2: ENCONTRAR GADGET DE 3 CORES (G)

Entrada: Grade parcial G

Saída: Gadget de 3 cores $G3$ caso exista

```
1  início
2      para cada v_i ∈ V(G) faça
3          enquanto não passar por todos P_4 induzidos que iniciar por v_i faça
4              Procure um P_4 induzido que inicia por v_i
5              se encontrou o P_4 então
6                  G3 ← PINTAR G3(P_4)
7                  retorna G3
8              fim
9          fim
10     fim
11     retorna NULO
12 fim
```

Fonte: Elaborado Pelo Autor

Algoritmo 3: PINTAR G3(P_4)

Entrada: P_4 que é *gadget* de 3 cores

Saída: Gadget de 3 cores

```
1  início
2      pinte P_4[1] e P_4[4] com cor 1.
3      pinte P_4[2] com cor 2.
4      pinte P_4[3] com cor 3.
5      V(G3) ← V(P_4)
6      E(G3) ← E(P_4)
7      retorna G3
8  fim
```

Fonte: Elaborado Pelo Autor

4.1.2 Encontrar gadget *de 4 cores*

O algoritmo 4 se propõe a encontrar um *gadget* de 4 cores $G4$ em uma grade parcial. O algoritmo procura por um P_6 candidato a *gadget* de 4 cores de forma enumerativas semelhante ao dito na subseção 4.1.1. Porém em vez de procurarmos por P_4, tentamos procurar P_6 candidato *gadget* de 4 cores que é um P_6 em que os dois vértices centrais tem grau pelo menos 3, o primeiro e o ultimo vértice não são vizinhos, o segundo e quinto vértice não são vizinhos.

Uma vez encontrado o P_6, passamos esse caminho para como parâmetro para *PINTAR G4*. Em *PINTAR G4* procuramos por dois vértice c_1 e d_1,onde c_1 é vizinho a $P_6[3]$ e não é vizinho de $P_6[1]$, e d_1 é vizinho a $P_6[4]$ e não é vizinho a $P_6[6]$. Se encontramos esses dois vértices, então pintamos eles com cor 1, e pintamos os vértices do P_6, e por fim, montamos o $G4$ e o retornamos.

Algoritmo 4: ENCONTRAR GADGET DE 4 CORES (G)

Entrada: Grade parcial G

Saída: Gadget de 4 cores $G4$ caso exista

```
1  início
2      para cada v_i ∈ V(G) faça
3          enquanto não passar por todos P_6 gadget de 4 cores que iniciar por v_i faça
4              Procure um gadget de 4 cores que inicia por v_i
5              se encontrou o P_6 então
6                  G4 ← PINTAR G4(P_6)
7                  se G4 ≠ NULO então
8                      retorna G4
9                  fim
10             fim
11         fim
12     fim
13     retorna NULO
14 fim
```

Fonte: Elaborado Pelo Autor

Algoritmo 5: PINTAR G4(P_6)

Entrada: P_6 que é *gadget* de 6 cores

Saída: Gadget de 3 cores

início

  Encontre dois vértices c_1 e d_1, em que $c_1 \in N(P_6[3])$, $d_1 \in N(P_6[4])$, $c_1 \notin N(d_1)$, $c_1 \notin N(P_6[1])$ e $d_1 \notin N(P_6[6])$.

  se *não* *Encontrou* c_1 *ou* ***não*** *Encontrou* d_1 **então**

    retorna $NULO$

  fim

  pinte c_1 e d_1 com cor 1

  pinte $P_6[1]$ e $P_6[6]$ com cor 1

  pinte $P_6[2]$ e $P_6[5]$ com cor 2

  pinte $P_6[3]$ com cor 3.

  pinte $P_6[4]$ com cor 4.

  $V(G4) \leftarrow V(P_6 \cup c_1 \cup d_1)$

  $E(G4) \leftarrow E(P_6 \cup c_1 \cup d_1)$

  retorna $G4$

fim

Fonte: Elaborado Pelo Autor

4.1.3 Encontrar gadget *de 5 cores*

O algoritmo 6 procura um *gadget* de 5 cores $G5$ em uma grade parcial G. Nesse algoritmo, procuramos por um P_4 que é candidato a *backbone*. Um P_4 é candidato a *backbone* se as suas extremidades têm grau pelo menos 3 e os dois vértices centrais tenham grau igual a 4. Uma vez encontrado esse P_4, iremos tentar montar um *gadget* de 5 cores.

Para isso, olharemos para os vizinhos de fora do P_4 de cada vértice no P_4 e tentamos escolher um vértice para ter a cor 2. O vértice que receber a cor 2 tem que ser vizinho de pelo menos um vértice que não esteja no *backbone* para ter a cor 1. Após feito isso, verificamos se para cada vértice no P_4 este tem um vizinho com cor 1 e outro com cor 2. Caso não exista, retornamos um valor nulo, e é procurado outro P_4 candidato a *backbone*. Caso contrário, pintamos os vértices do P_4 e os vértices em $N^2(P_4)$, por fim montamos o $G5$ e o retornamos.

Fonte: Elaborado Pelo Autor

Algoritmo 6: ENCONTRAR GADGET DE 5 CORES (G)

Entrada: Grade parcial G
Saída: Gadget de 5 cores $G5$ caso exista

```
1  início
2      para cada v_i ∈ V(G) faça
3          para cada P_4 candidato a backbone que iniciar por v_i faça
4              se encontrou o P_4 então
5                  G5 ←PINTAR G5(P_4)
6                  se G5 ≠ NULO então
7                      retorna G5
8                  fim
9              fim
10         fim
11     fim
12     retorna NULO
13 fim
```

Algoritmo 7: PINTAR G5(P_4)

Entrada: Um caminho P_4

Saída: Gadget de 5 cores $G5$ caso exista

início
  pinte os dois vértices da ponta do P_4 com cor 3
  pinte os dois vértices centrais do P_4, um com cor 4 e o outro com cor 5.
  para cada $v_i \in P_4$ **faça**
    pegue um vértice com grau maior que 1 em $N(v_i) - P_4$ que não tem cor e não é vizinho de algum vértice com cor 2, e possui um vizinho que não está em P_4, e pinte com 2.
    pegue um vértice em $N(v_i) - P_4$ que não tem cor e não é vizinho de algum vértice com cor 1, e pinte com 1.
  fim
  para cada $v_i \in P_4$ **faça**
    se *v_i não tem nenhum vizinho com cor 1* ***ou*** *v_i não tem nenhum vizinho com cor 2* **então**
      retorna *NULO*
    fim
  fim
  para cada $v_i \in N^2(P_4)$ **faça**
    se *v_i não possui cor* ***e*** *é vizinho de algum vértice com cor 2* ***e*** *não é vizinho de algum vértice com cor 1* **então**
      pinte v_i com cor 1.
    fim
  fim
  $V(G5) \leftarrow V(P_4) \cup V(N^1(P_4)) \cup V(N^2(P_4))$
  $E(G5) \leftarrow E(P_4) \cup E(N^1(P_4)) \cup E(N^2(P_4))$
  retorna $G5$
fim

Fonte: Elaborada Pelo Autor

4.1.4 Estratégia 1

A Estratégia 1 utiliza os algoritmos 6,4 e 2, de forma que ele tenta verificar primeiramente se o grafo G passado possui $\Gamma(G) = 5$; se possui então retorna *gadget* de 5 cores; se não o algoritmo vai verificar se o grafo tem $\Gamma(G) = 4$, retornando o *gadget* de 4 cores encontrado em caso positivo. Caso não encontre ele irá finalmente verificar se o grafo possui $\Gamma(G) = 3$ e retornar um *gadget* de 3 cores ou *NULO* a depender se foi possível ou não encontrá-lo.

Essa estratégia tem esse comportamento pois, na ordem testada, o primeiro tipo de *gadget* encontrado determina o número de Grundy do grafo de maneira direta, dados os teoremas apresentados, já que a inexistência de um *gadget* de i cores significa que $\Gamma(G) < i$, considerando $i \in \{1,2,3,4,5\}$.

4.1.5 Estratégia 2

Na Estratégia 2, para cada vértice de G tentamos montar de forma enumerativa um P_6 que possa ser candidato para montar um *gadget* de 4 cores.

Se no processo de enumeração encontramos um P_4 induzido, que estará como "prefixo" de algum dos P_6 a enumerar, o algoritmo monta o *gadget* de 3 cores com esse P_4, e passa a verificar se este P_4 pode ser um *backbone* de um *gadget* de 5 cores. Caso seja, então é feito o processo de pintar que retorna o *gadget* de 5 cores caso exita. Caso contrário, continuamos tentando montar o P_6.

Caso o algoritmo consiga encontrar algum P_6, montamos um *gadget* de 4 cores. Em seguida é verificado se os 4 vértices centrais desse P_6 podem ser um *backbone*, caso seja tentamos colorir, e retornamos o *gadget* de 5 cores caso exista.

Uma vez encontrado um *gadget* de 4 cores, o algoritmos apenas se preocupará em encontrar apenas P_4s que são candidatos a *backbone*.

Após todo esse processo se não encontrou o *gadget* de 5 cores, é verificado se encontrou um *gadget* de 4 cores, se sim é retornado este *gadget*. Caso contrario é verificado se encontrou o *gadget* de 3 cores, se sim retorna esse *gadget*. Caso contrario, retorna nulo.

Essa estratégia procede dessa forma pois, pelas definições dadas, qualquer *gadget* de 4 cores necessariamente contém um *gadget* de 3 cores e qualquer gadget de 5 cores necessariamente contém um *gadget* de 4 cores. Além disso, ao encontrarmos o primeiro *gadget* para uma quantidade de cores, não necessitamos mais continuar a busca por esse tipo.

4.2 Estudos de caso

Nesta seção, descrevemos os experimentos computacionais realizados de forma a avaliarmos a execução das estratégias propostas. De forma a parametrizarmos os algoritmos com algum referencial, propomos um experimento posterior utilizando o algoritmo guloso sobre os grafos usados para os testes.

Dados os teoremas apresentados neste trabalho e o funcionamento das estratégias propostas, podemos deduzir que ambas fornecem de forma correta a informação de qual é o número de Grundy de uma dada grade parcial sempre que este for pelo menos 3, já que os demais casos são trivialmente verificados. Dessa forma, de posse do valor ótimo, decidimos verificar o tempo que o algoritmo guloso leva para encontrar uma coloração gulosa com o número ótimo de cores.

Para esse experimento, dada a mesma ordem total para os vértices de G considerada pelas estratégias propostas, informamos ao algoritmo 1 uma permutação dessa ordem a cada iteração, em ordem lexicográfica. O tempo marcado é o tempo decorrido até alcançarmos uma permutação para a qual a resposta do algoritmo guloso seja uma coloração com o mesmo número de cores que o que as nossas estratégias já determinaram ser a quantidade ótima. A esse algoritmo denominamos "Algoritmo Guloso Lexicográfico".

Perceba que o tempo obtido não significa o tempo de resolução do problema pelo algoritmo guloso, já que este só é capaz de fornecer limites inferiores para o número de Grundy por definição. Além disso, a forma em que as permutações são informadas ao algoritmo é apenas uma forma simples e conveniente, o que não significa que seja a melhor escolha. Esses resultados são, portanto, apenas um referencial de tempo para quão rápido é o guloso em encontrar uma solução ótima, nesses termos, sem entretanto provar sua otimalidade.

4.2.1 Algoritmo gerador de grafos aleatórios

Os experimentos realizados se baseiam em grades parciais geradas aleatoriamente. Nosso algoritmo de criação de grafos aleatórios para o problema em estudo foi criado baseado no algoritmo 8, apresentado no trabalho de Bastos (2012) onde temos que o algoritmo que gera um grafo simples aleatório G em que n é o número de vértices e p como sendo a probabilidade de ocorrer aresta entre dois vértices. Quando maior é o valor de p, maior é a probabilidade de ocorrer arestas em G. Por isso temos que p influenciar a quantidade de arestas que podem

ocorrer em G.

Algoritmo 8: GERADOR DE GRAFOS ALEATÓRIOS(N, P)

```
1  início
2      V(G) ← {1,...,n}
3      E(G) ← ∅
4      para i = 1 → n − 1 faça
5          para j = i + 1 → n faça
6              se rand() <= p então
7                  E(G) ← E(G) ∪ {(i, j)}
8              fim
9          fim
10     fim
11     retorna G
12 fim
```

Fonte: Baseado no trabalho de (BASTOS, 2012)

Considere que $rand()$ é uma função que retorna um valor aleatório uniforme dentro do intervalo $[0.0, 1.0]$ a cada chamada.

O princípio utilizado no algoritmo 8 funciona para grafos simples em geral, onde não haja restrição para o número de vizinhos de um vértice. Porém, em nosso trabalho estamos estudando grades parciais e esses tipos de grafos tem um limite superior para o número de vizinhos de um vértice. Sendo assim, foi feito o algoritmo 9 em que passamos como entrada N e M como se fossem dimensões de uma grade, onde N representaria as linhas e M as colunas, e p como sendo a probabilidade de uma aresta acontecer entre dois vértices. Os vértices nesse algoritmo são enumerados de 0 até $(N \cdot M) - 1$. O algoritmo deverá, então, observar cada par de vértices que poderia ser adjacente em uma grade e determinar se essa aresta existirá ou não.

Para sabermos se pode existir aresta entre dois vértices do nosso grafo é feita uma verificação onde observamos se um vértice v está na ultima coluna dessa distribuição matricial através da função *borda direita*, onde passamos v como parâmetro, e testamos se $(v+1)$ mod $M = 0$. Caso seja verdade, não pode haver arestas entre o vértice v e o vértice $v+1$, pois este último estará no início da próxima linha (ou talvez não exista, no vaso de $v = N \cdot M - 1$).

Também é verificado se um vértice v não está na ultima linha utilizando a função *borda inferior*, onde passamos como parâmetro v e testamos se v div $M = N - 1$. Caso seja verdade não pode haver arestas entre os vértices v e $v + M$. Desse modo um vértice no grafo

só pode ter um vizinho que está na linha acima, um vizinho que esta na linha de abaixo, um vizinho na coluna a direita e um vizinho na coluna a esquerda, e para cada uma dessas posições determinamos as que existem e escolhemos aleatoriamente a inserção da aresta com a probabilidade escolhida.

Algoritmo 9: GERADOR DE GRADES PARCIAIS ALEATÓRIAS(N, M, P)

Entrada: N, M, p

Saída: G

```
1  início
2      n ← N × M
3      V(G) ← {0, ..., n − 1}
4      E(G) ← ∅
5      para v ∈ V(G) faça
6          se ! borda direita(v) então
7              se rand() <= p então
8                  E(G) ← E(G) ∪ {(v, v + 1)}
9              fim
10         fim
11         se ! borda inferior(v) então
12             se rand() <= p então
13                 E(G) ← E(G) ∪ {(v, v + M)}
14             fim
15         fim
16     fim
17     retorna G
18 fim
```

Fonte: Baseado no trabalho de (BASTOS, 2012)

4.2.2 Experimentos

Os algoritmos utilizados nos experimentos foram feitos na linguagem de programação *C++*, e foram executados em um computador com processador Intel(R) Core(TM) i5 - 8250U CPU @ 1.60GHz, 8 GB de memória RAM e Windows 64 bits. Em nosso experimento foram gerados grafos aleatórios, utilizando probabilidade de geração de arestas, de 30, 50, 70 e

90 por cento, onde para cada probabilidade foram gerados 5 grafos com 20, 40, 60, 80, 100, 150 e 200 vértices, totalizando ao todo 140 grafos.

Devido a falta de trabalhos que apresentassem alguma heurística para o problema em estudo, foram feitos testes apenas com os algoritmos apresentado neste trabalho.

Para o experimento foram utilizados os Algoritmos *Estratégia 1*, *Estratégia 2* e *Guloso Lexicográfico*. Utilizamos o tempo máximo de 600 segundos (10 minutos) para testar o tempo que cada algoritmo leva para encontrar o valor de $\Gamma(G)$.

A Tabela 3 no apêndice A apresenta o tempo em segundos de execução de que cada algoritmo levou pra encontrar o valor de $\Gamma(G)$ por número de vértices, arestas e densidade. Se o tempo de execução estiver superior a 600 segundos significa que a execução do algoritmo foi interrompido já que ultrapassou o tempo limite.

Como podemos verificar na Tabela 3 do apêndice A, os Algoritmos *Estratégia 1* e *Estratégia 2*, em todos os testes, conseguiram gerar uma resposta em menos de um segundo. Já o *Algoritmo Guloso Lexicográfico* em grande parte dos testes não conseguiu obter uma resposta pois ultrapassou o tempo limite de 600 segundos.

O Algoritmo *Estratégia 2* mostrou ter um desempenho melhor que o algoritmo *Estratégia 1* quando $\Gamma(G) = 3$. O tempo de execução em média do *Estratégia 2* ficou abaixo de 0.0015 segundos e o do *Estratégia 1* ficou acima dos 0.0035 segundos. No gráfico da Figura 13, no apêndice B, é mostrada uma noção gráfica da diferença de desempenho entre os algoritmos para $\Gamma(G) = 3$. E também pode ser observado essa diferença na Tabela 3 do apêndice A.

Para $\Gamma(G) = 4$, em quase todos os testes o algoritmo *Estratégia 2* mostrou ser superior ao *Estratégia 1*. Como pode ser observado na Tabela 3, e no gráfico da Figura 14 do apêndice B, o *Estratégia 2* conseguiu executar em muitos casos em menos de 0.002 segundos, já o *Estratégia 1* ficou superior a esse tempo em todos os casos, para $\Gamma(G) = 4$

Já para $\Gamma(G) = 5$, o Algoritmo *Estratégia 1* se mostrou ter um desempenho melhor que *Estratégia 2* sendo que a diferença de desempenho em muitos casos é muito pequena entre esses dois algoritmos, como podemos observar na Tabela 3 do apêndice A e no gráfico da Figura 15 do apêndice B.

O *Algoritmo Guloso Lexicográfico* em alguns casos se mostrou ter um desempenho superior ao Algoritmo *Estratégia 1* e ao Algoritmo *Estratégia 2* para $\Gamma(G) = 3$ e $\Gamma(G) = 4$. Porém, foram poucos os casos que isso ocorreu, e em grande maioria o *Algoritmo Guloso Lexicográfico* não conseguiu determinar uma resposta no tempo limite. E para $\Gamma(G) = 5$ o *Algoritmo Guloso Lexicográfico* não conseguiu em nenhum teste obter a resposta em menos de

600 segundos. E isso pode ser observado na Tabela 3 do apêndice A.

A Tabela 1 mostra o tempo médio de execução em segundos por $\Gamma(G)$ para os Algoritmos *Estratégia 1* e *Estratégia 2*. Já a Tabela 2 mostra o tempo de médio de execução em segundos dos testes para os quais o *Algoritmo Guloso Lexicográfico* conseguiu uma resposta.

Tabela 1 – TEMPO MÉDIO EM SEGUNDOS DAS ESTRATÉGIAS - POR $\Gamma(G)$.

Numero Guloso $\Gamma(G)$	Estratégia 1	Estratégia 2
3	0.004061	0.000889
4	0.003016	0.001447
5	0.003240	0.004109

Fonte: Elaborado Pelo Autor
Nota: Cada linha representa o tempo médio em segundos para cada $\Gamma(G)$.

Tabela 2 – TEMPO MÉDIO DO ALGORITMO GULOSO LEXICOGRÁFICO - POR $\Gamma(G)$.

Número Guloso $\Gamma(G)$	Guloso Lexicográfico
3	3.6558
4	67.4161

Fonte: Elaborado Pelo Autor
Nota: Cada linha representa o tempo médio em segundos para cada $\Gamma(G)$.

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foram implementados 3 algoritmos que se propõem a determinar o valor de $\Gamma(G)$. Em específico os algoritmos *Estratégia 1* e o *Estratégia 2* se mostraram superiores ao *Algoritmo Guloso lexicografo* em grande parte dos testes. O que já era esperado, já que o *Algoritmo Guloso Lexicográfico* é exponencial no pior caso.

Nos casos $\Gamma(G) = 3$ e $\Gamma = 4$ o Algoritmo *Estratégia 2* teve a vantagem sobre os demais algoritmos para grande parte dos testes. Essa vantagem se dá pelo fato de como esse algoritmo é construído, já que no processo de verificar se uma grade parcial possui $\Gamma(G) = 4$ o algoritmo também verifica se a grade possui $\Gamma(G) = 3$ e $\Gamma(G) = 5$. Já o Algoritmo *Estratégia 1* teve vantagem em todos os testes para $\Gamma(G) = 5$. E isso ocorre porque esse algoritmo procura primeiro por $\Gamma(G) = 5$, caso não exista ele procura por $\Gamma(G) = 4$, e o $\Gamma(G) = 3$.

As sugestões para trabalhos futuros são de tentar encontrar uma forma mais eficiente de buscar os *gadgets*, através de uma redução desse problema a outro, como por exemplo tentar reduzir a uma versão do problema de emparelhamento no caso do *gadget* de 5 cores, ou utilizar fórmulas lógicas e *SAT solvers*, ou mesmo tentar elaborar uma heurística. Além disso, instâncias de testes também podem ser maiores e mais detalhadas.

REFERÊNCIAS

BASTOS, L. **Novos algoritmos e resultados teóricos para o problema de particionamento de grafos por edição de arestas**. Tese (Doutorado) — Rio de Janeiro: Universidade Federal Fluminense, 2012.

BONDY, J. A.; MURTY, U. S. R. *et al.* **Graph theory with applications**. [S.l.]: Citeseer, 1976. v. 290.

EFFANTIN, B.; KHEDDOUCI, H. Grundy number of graphs. **Discussiones Mathematicae Graph Theory**, De Gruyter Open, v. 27, n. 1, p. 5–18,[S.l], 2007.

FERREIRA, T. de O. **Conjunto independente e cobertura por cliques em grafos de disco unitário de moeda unitária**. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2003.

GAMA, S. I. M. *et al.* **Sobre Problemas de Lista Coloração e a propriedade de selecionabilidade em grafos**. [S.l.]: Universidade Federal do Amazonas, 2016.

NETO, A. S. A.; GOMES, M. J. N. Problema e algoritmos de coloracao em grafos-exatos e heuristicos. **Revista de Sistemas e Computação-RSC**, v. 4, n. 2, [S.l], 2015.

OLIVEIRA, A. K. Maia de. **Estudo de Casos de Complexidade de Colorações Gulosa de Vértices e de Arestas**. [S.l.]: Universidade Federal do Ceará, 2011.

SAMPAIO, L. **Algorithmic aspects of graph colourings heuristic**. Tese (Doutorado) — [S.l.:s.n], 2012.

SILVA, T. G. N. da; ROCHA, L. S.; VALDISIO, G.; VIANA, R. Proposta e avaliação de novas heurísticas para o problema de coloração de vértices. **Programação XLVIII SBPO**, [S.l], 2016.

SOUZA, B. S. d. **Produtos e coespectralidade de grafos**. [S.l.]: Universidade Federal do Rio Grande do Sul, 2016.

WEST, D. B. *et al.* **Introduction to graph theory**. [S.l.]: Prentice hall Upper Saddle River, NJ, 1996. v. 2.

APÊNDICE A – TABELA DE COMPARAÇÃO DE DESEMPENHO

Na tabela abaixo são exibido os resultados obtidos com os testes feitos com os algoritmos descritos neste trabalho. Na primeira coluna é mostrado a quantidade de vértices que o grafo testado possui, na segunda é mostra o numero de arestas que o grafo testado possui, na terceira coluna é mostrada a probabilidade de existência de cada aresta por cento utilizada na criação do grafo, na quarta coluna é exibido o número guloso que possível atribuir ao grafo, na quinta coluna é exibido o tempo em segundos que o algoritmo *Estratégia 1* levou para encontrar o número guloso do grafo, na sexta coluna é exibido o tempo em segundos que o algoritmo *Estratégia 2* levou para encontrar o número guloso do grafo e por fim na sétima coluna é o tempo em segundos que o algoritmo *Guloso lexicográfico* levou para encontrar o número guloso do grafo. Em cada teste foi dado o tempo limite de 600 segundos, nas células que são apresentado o tempo de execução dos algoritmos, se ela possui um tempo maior que 600 segundos isso significa que a execução do algoritmo foi interrompida antes de conseguir encontrar o número guloso para aquele teste.

Tabela 3 – TEMPO DE EXECUÇÃO EM SEGUNDOS DOS ALGORITMOS - POR NÚMERO DE VÉRTICES, ARESTAS, DENSIDADES E $\Gamma(G)$

N.vértices	N.arestas	Probabilidade(%)	$\Gamma(G)$	Estratégia 1	Estratégia 2	Guloso lexicográfico
20	13	30	3	0.00398	0.001072	0.0000038
20	5	30	2	0.004805	0.00005	0.0000024
20	8	30	3	0.003939	0.000929	0.0000029
20	7	30	3	0.003888	0.000728	69.448
20	12	30	3	0.004267	0.00067	0.0000036
20	13	50	3	0.003962	0.000666	0.0000032
20	19	50	4	0.002593	0.000989	0.0004887
20	13	50	4	0.002874	0.001228	0.0135597
20	17	50	3	0.004185	0.000685	0.0124412
20	18	50	4	0.00261	0.000678	0.0319756
20	23	70	4	0.003718	0.003082	362.688
20	25	70	5	0.004768	0.005488	600.001
20	16	70	3	0.004272	0.000643	0.0000079
20	23	70	4	0.002605	0.0011	0.284951
20	23	70	4	0.002753	0.002	0.012857
20	29	90	5	0.003894	0.008735	600.001
20	29	90	5	0.002239	0.003575	600.001
20	29	90	5	0.00101	0.001286	600.001

N.vértices	N.arestas	Probabilidade(%)	$\Gamma(G)$	Estratégia 1	Estratégia 2	Guloso lexicográfico
20	28	90	5	0.002194	0.002933	600.001
20	30	90	5	0.006128	0.00657	600.001
40	27	30	4	0.002722	0.001124	600.001
40	25	30	4	0.002906	0.00123	0.0000308
40	26	30	3	0.004241	0.00096	0.0000054
40	17	30	3	0.00412	0.00088	0.0000044
40	27	30	3	0.003911	0.000758	0.0000057
40	36	50	4	0.002966	0.001295	600.001
40	36	50	4	0.0028	0.001125	600.001
40	34	50	4	0.002903	0.000886	0.0002766
40	30	50	4	0.002726	0.001216	600.001
40	31	50	4	0.003082	0.00088	600.001
40	50	70	4	0.005616	0.00438	6.72838
40	47	70	4	0.002565	0.001048	600.001
40	51	70	5	0.001131	0.002964	600.001
40	49	70	4	0.003118	0.003509	0.0004197
40	51	70	5	0.002086	0.002977	600.001
40	63	90	5	0.003601	0.004378	600.001
40	62	90	5	0.005137	0.005894	600.001
40	58	90	5	0.004037	0.004691	600.001
40	59	90	5	0.004109	0.005964	600.001
40	63	90	5	0.00358	0.004896	600.001
60	27	30	3	0.004011	0.001003	0.0000069
60	29	30	3	0.004114	0.000885	600.001
60	28	30	3	0.004316	0.000965	0.0000141
60	35	30	3	0.004337	0.000938	0.0000168
60	36	30	4	0.002894	0.000997	600.001
60	51	50	4	0.002665	0.000882	600.001
60	53	50	5	0.000973	0.001446	600.001
60	52	50	4	0.002994	0.00111	600.001
60	54	50	4	0.002962	0.001983	600.001
60	53	50	4	0.002885	0.001261	0.0000457
60	75	70	5	0.006157	0.007348	600.001
60	75	70	5	0.001054	0.001891	600.001
60	67	70	5	0.00195	0.002195	600.001
60	80	70	5	0.00186	0.002065	600.001

N.vértices	N.arestas	Probabilidade(%)	$\Gamma(G)$	Estratégia 1	Estratégia 2	Guloso lexicográfico
60	65	70	4	0.002948	0.001057	600.001
60	91	90	5	0.006044	0.006367	600.001
60	94	90	5	0.003173	0.00355	600.001
60	93	90	5	0.01297	0.014092	600.001
60	94	90	5	0.006428	0.007712	600.001
60	87	90	5	0.003136	0.003526	600.001
80	49	30	4	0.002811	0.001112	600.001
80	44	30	4	0.00301	0.001001	600.001
80	49	30	4	0.002877	0.000974	600.001
80	36	30	4	0.002667	0.001049	600.001
80	49	30	4	0.002732	0.000892	600.001
80	71	50	4	0.002699	0.000926	600.001
80	65	50	4	0.002935	0.00106	600.001
80	70	50	4	0.002844	0.001143	600.001
80	71	50	5	0.002985	0.003513	600.001
80	73	50	4	0.003054	0.001007	600.001
80	105	70	5	0.001697	0.002304	600.001
80	103	70	5	0.000995	0.001381	600.001
80	103	70	5	0.001134	0.001673	600.001
80	96	70	5	0.001037	0.001367	600.001
80	105	70	5	0.001014	0.002252	600.001
80	133	90	5	0.004038	0.00492	600.001
80	132	90	5	0.000959	0.00147	600.001
80	129	90	5	0.006263	0.006951	600.001
80	132	90	5	0.001924	0.002542	600.001
80	132	90	5	0.001878	0.003085	600.001
100	51	30	4	0.002719	0.001062	600.001
100	50	30	4	0.002604	0.001131	600.001
100	63	30	4	0.002924	0.001169	600.001
100	49	30	3	0.003712	0.000898	0.0000097
100	61	30	4	0.002946	0.001075	600.001
100	94	50	4	0.002822	0.001363	600.001
100	85	50	4	0.002886	0.001016	506.649
100	93	50	5	0.004006	0.004707	600.001
100	90	50	5	0.001235	0.002047	600.001
100	82	50	4	0.002987	0.000916	0.0000402

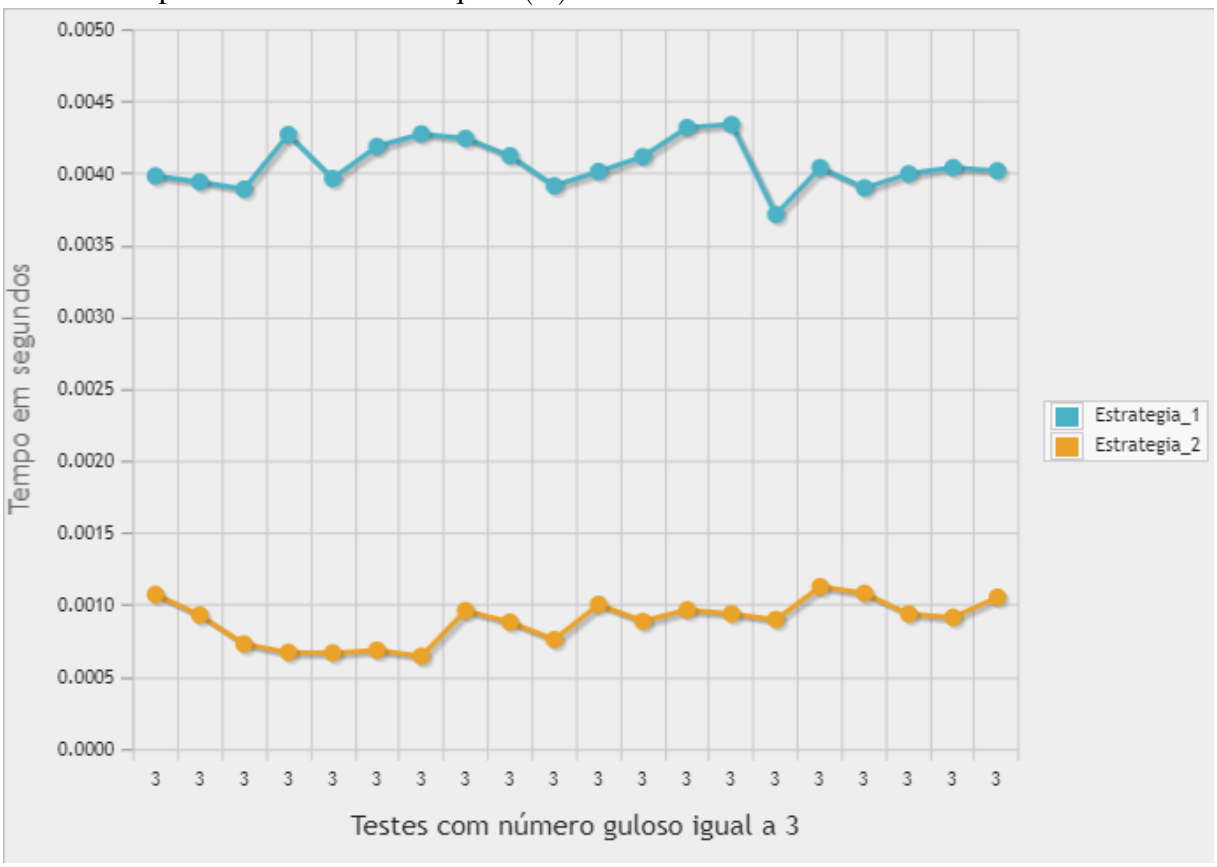
N.vértices	N.arestas	Probabilidade(%)	$\Gamma(G)$	Estratégia 1	Estratégia 2	Guloso lexicográfico
100	122	70	5	0.000996	0.001175	600.001
100	135	70	5	0.003887	0.004395	600.001
100	124	70	5	0.002052	0.002399	600.001
100	137	70	5	0.00107	0.0014	600.001
100	128	70	5	0.00357	0.004579	600.001
100	163	90	5	0.001737	0.002671	600.001
100	166	90	5	0.014848	0.016085	600.001
100	163	90	5	0.003901	0.004533	600.001
100	162	90	5	0.003872	0.004449	600.001
100	160	90	5	0.002114	0.00244	600.001
150	87	30	4	0.003179	0.001113	600.001
150	77	30	4	0.003104	0.001174	600.001
150	84	30	3	0.004038	0.001126	0.0000149
150	83	30	4	0.003022	0.001145	600.001
150	76	30	3	0.003897	0.00108	0.0000139
150	142	50	5	0.000923	0.001279	600.001
150	142	50	4	0.004744	0.005573	600.001
150	147	50	4	0.002916	0.001414	600.001
150	136	50	4	0.002978	0.001246	600.001
150	136	50	4	0.004709	0.003615	600.001
150	189	70	5	0.00281	0.003613	600.001
150	181	70	5	0.003225	0.004388	600.001
150	202	70	5	0.001968	0.002029	600.001
150	211	70	5	0.002068	0.002598	600.001
150	194	70	5	0.002275	0.004011	600.001
150	244	90	5	0.003117	0.004658	600.001
150	242	90	5	0.001021	0.001331	600.001
150	245	90	5	0.007219	0.009474	600.001
150	249	90	5	0.003985	0.00449	600.001
150	244	90	5	0.002095	0.002595	600.001
200	101	30	3	0.003995	0.000937	0.0000189
200	113	30	3	0.004038	0.000913	0.0000194
200	112	30	4	0.002964	0.001404	600.001
200	119	30	3	0.004016	0.001053	0.0000196
200	125	30	4	0.003088	0.001092	600.001
200	173	50	4	0.003042	0.001507	600.001

N.vértices	N.arestas	Probabilidade(%)	$\Gamma(G)$	Estratégia 1	Estratégia 2	Guloso lexicográfico
200	179	50	4	0.002937	0.002153	600.001
200	206	50	5	0.002785	0.004642	600.001
200	191	50	5	0.001858	0.002377	600.001
200	186	50	4	0.002713	0.000973	600.001
200	256	70	5	0.007498	0.010039	600.001
200	258	70	5	0.001124	0.00161	600.001
200	243	70	5	0.003982	0.005792	600.001
200	269	70	5	0.00416	0.005279	600.001
200	257	70	5	0.001978	0.002356	600.001
200	323	90	5	0.004999	0.005885	600.001
200	332	90	5	0.001965	0.002693	600.001
200	336	90	5	0.002973	0.003302	600.001
200	338	90	5	0.003022	0.00374	600.001
200	330	90	5	0.0017	0.002525	600.001

APÊNDICE B – GRÁFICOS DE COMPARAÇÃO DE TEMPO DE EXECUÇÃO

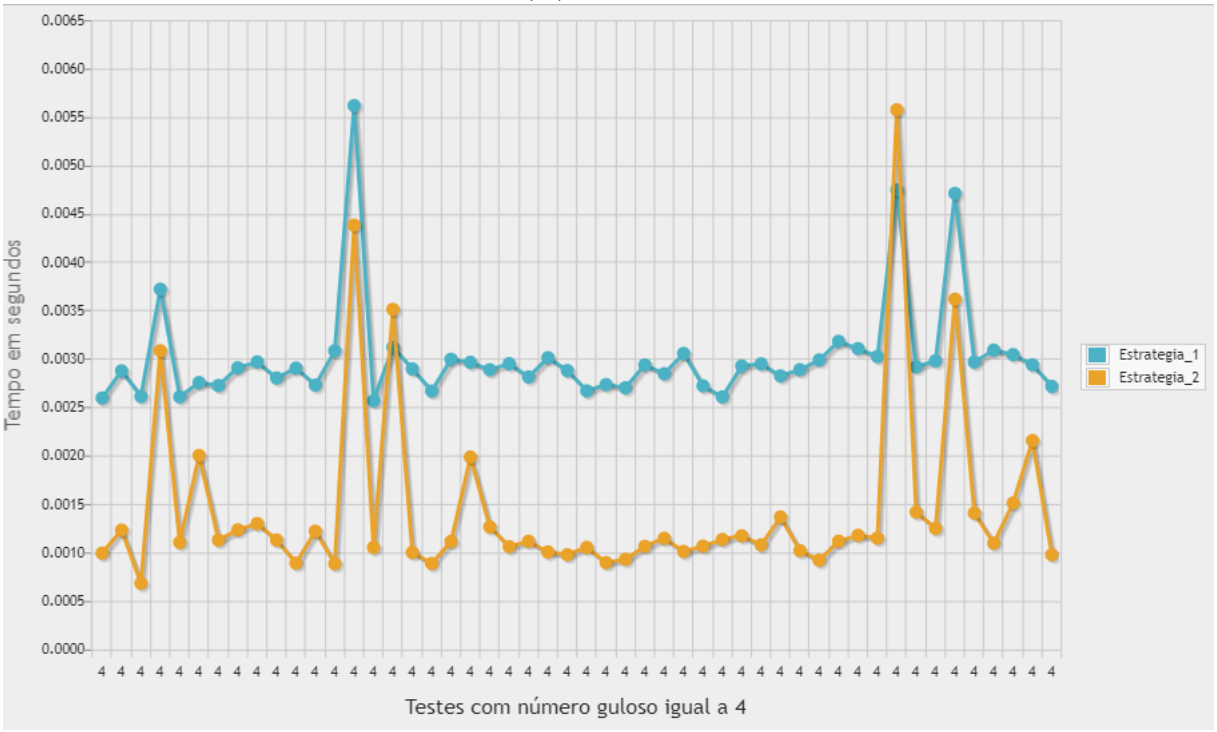
As figuras abaixo mostram graficamente uma comparação entre o tempo de execução dos algoritmos *Estratégia 1* e *Estratégia 2*. Na Figura 13 mostra a comparação para todos os teste em que $\Gamma(G) = 3$, já na Figura 14 mostra a comparação para todos os teste em que $\Gamma(G) = 4$, e por fim na Figura 15 mostra a comparação para todos os teste em que $\Gamma(G) = 5$.

Figura 13 – Gráfico comparando o tempo de execução dos algoritmos Estratégia 1 , Estratégia 2 por todos os testes em que $\Gamma(G) = 3$



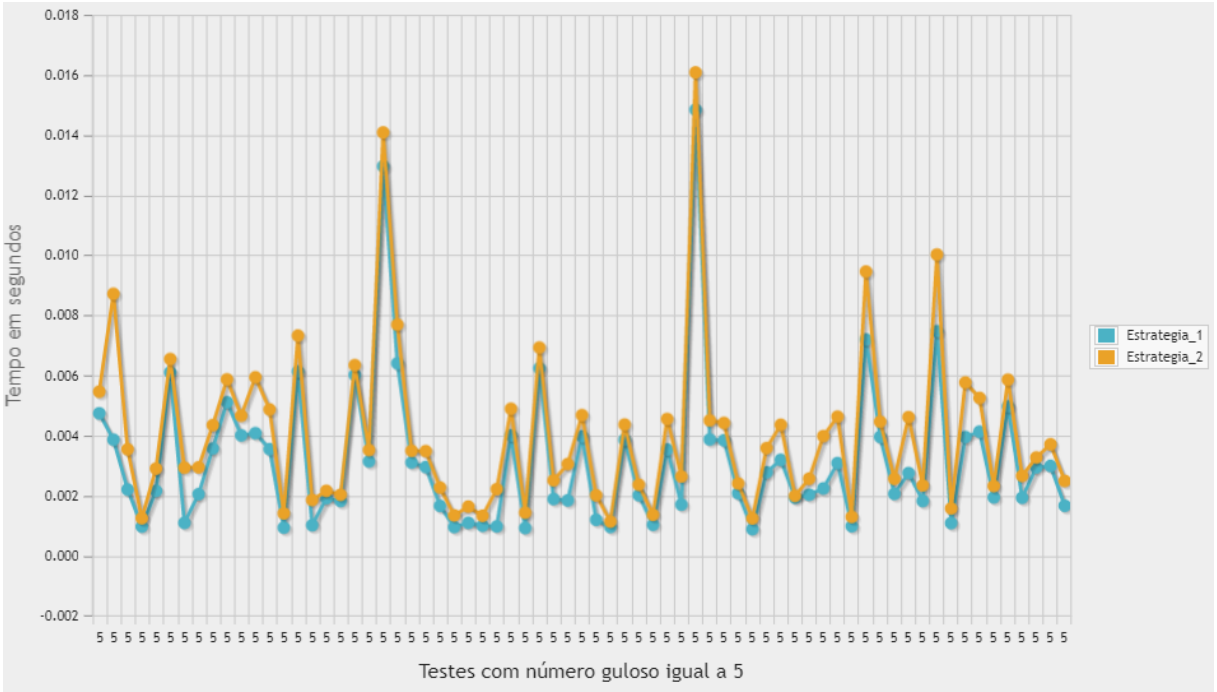
Fonte: Elaborada pelo Autor

Figura 14 – Gráfico comparando o tempo de execução dos algoritmos Estratégia 1, Estratégia 2 por todos os testes em que $\Gamma(G) = 4$



Fonte: Elaborada pelo Autor

Figura 15 – Gráfico comparando o tempo de execução dos algoritmos Estratégia 1, Estratégia 2 por todos os testes em que $\Gamma(G) = 5$



Fonte: Elaborada pelo Autor