Geração de Obstruções Minimais de Grafos-$(2,1)$

Matheus Souza D'Andrea Alves
Universidade Federal Fluminense - UFF
Niterói - RJ
mad@id.uff.br

Raquel Bravo
Universidade Federal Fluminense - UFF
Niterói - RJ
raquelbr.ic@gmail.com

Loana T. Nogueira
Universidade Federal Fluminense - UFF
Niterói - RJ
loanatito@gmail.com

Uéverton dos Santos Souza
Universidade Federal Fluminense - UFF
Niterói - RJ
ueverton@ic.uff.br

RESUMO

Um grafo é dito um grafo-(k,ℓ) se o seu conjunto de vértices pode ser particionado em k conjuntos independentes e ℓ cliques. Reconhecer se um dado grafo G é um grafo-(k,ℓ) é um problema NP-completo para todo par k,ℓ tal que $\max\{k,\ell\} \geq 3$. Por outro lado, reconhecer se um dado G é um grafo-$(2,1)$ é uma tarefa executável em tempo polinomial. Embora o reconhecimento de grafos-$(2,1)$ seja polinomial, pouco se sabe sobre as características da família dos subgrafos-proibidos dos grafos-$(2,1)$, também conhecida como a família das obstruções minimais de grafos-$(2,1)$. Sendo assim, o objetivo deste trabalho é demonstrar algumas propriedades inerentes das obstruções de grafos-$(2,1)$, e em seguida fazer uso destas para desenvolver um algoritmo eficiente de enumeração das obstruções minimais de grafos-$(2,1)$ com até nove vértices, facilmente extensível para grafos de ordem maior.

Palavras Chave: Grafos-(k,ℓ), subgrafos proibidos, obstruções minimais, grafos-(2,1).
Área principal: TAG

ABSTRACT

A graph is called a (k,ℓ)-graph if its set of vertices can be partitioned into k independent sets and ℓ cliques. Recognize whether a given graph G is a (k,ℓ)-graph is an NP-complete problem for every pair k,ℓ such that $\max\{k,\ell\} \geq 3$. On the other hand, to recognize whether a given G is a $(2,1)$-graph can be done in polynomial time. Although the recognition of $(2,1)$-graphs is polynomial-time resolvable, there is not much that is known about the family of forbidden subgraphs of $(2,1)$-graphs, also known as minimal obstructions of $(2,1)$-graphs. Thus, the aim of this study is to demonstrate some inherent properties of obstructions of $(2,1)$-graphs, and then make use of such properties to develop an efficient algorithm to enumerate the minimal obstructions of $(2,1)$-graphs with at most nine vertices, which can easily be extended to work with larger graphs.

Keywords: (k,ℓ)-graphs, forbidden subgraphs, minimal obstructions, $(2,1)$-graphs.
Main area: Graph Theory

1. Introdução

A teoria dos grafos tem aplicabilidades em diversas áreas (Matemática, Informática, Engenharia, Biologia, Indústria, dentre outras), pois um grafo constitui o modelo matemático ideal para o estudo das relações entre objetos discretos de qualquer tipo. Em particular, grafos têm sido úteis na definição e/ou resolução de problemas da vida real que vão desde problemas de localização, traçados de rotas, projetos de informática, até o seu uso em ciências sociais. Sua importância cresceu muito no século XX, com o surgimento das redes de energia elétrica e de telecomunicação, dos circuitos digitais e, por fim, dos computadores.

Em Ciência da Computação, grafos podem ser usados como tipos abstratos de dados, em planejamento eficiente para determinar as rotas de transmissão de pacotes na internet, organização de dados, inteligência artificial e produção de algoritmos para solução de problemas levando em consideração a execução em tempo razoável. É possível obter resultados importantes, pois existem vários problemas computacionais que empregam grafos com sucesso. No entanto, para muitos problemas em grafos não são conhecidos algoritmos eficientes para sua solução.

Uma das principais motivações do estudo de classes de grafos é o fato de que diversos problemas, que são difíceis para grafos em geral, tornam-se tratáveis quando restritos a classes especiais de grafos. Assim, busca-se delimitar a partir de que ponto um determinado problema pode ser resolvido de forma eficiente. Particularmente, o problema de particionamento em grafos tem despertado muito interesse devido às pesquisas de grafos perfeitos [Golumbic (1980)] e também pela procura de algoritmos eficientes para o reconhecimento de determinadas classes de grafos. São exemplos de algumas aplicações que podem ser resolvidas através de problemas de particionamento: segmentação de imagens, encontrar subconjuntos de páginas da Web que estão relacionadas entre si, o problema da k-coloração (de grande importância na teoria dos grafos) com aplicações em questões práticas como escalonamento de tarefas e alocação de freqüências. Outro problema também conhecido de particionamento de grafos é verificar se um dado grafo G é split, ou equivalentemente, verificar se o conjunto dos vértices de G pode ser particionado em dois subconjuntos, dos quais um é independente e o outro é uma clique. Provou-se que o reconhecimento de grafos split pode ser realizado em tempo linear [Golumbic (1980)].

Formalmente, muitos desses problemas (de partição de grafos) podem ser descritos como tendo por objetivo particionar o conjunto dos vértices de um grafo em subconjuntos $V_1, V_2,$ $\ldots, V_k$ onde $V_1 \cup V_2 \cup \ldots \cup V_k = V$ e $V_i \cap V_j = \emptyset$, $i \neq j$, $1 \leq i \leq k$ e $1 \leq j \leq k$, exigindo-se, porém, algumas propriedades sobre estes subconjuntos de vértices. Estas propriedades podem ser *internas*, como por exemplo exigir que os vértices de cada subconjunto V_i sejam dois-a-dois adjacentes (isto é, V_i é uma clique) ou dois-a-dois não-adjacentes (isto é, V_i é um conjunto independente), ou *externas*, onde as exigências são feitas sobre os pares de $V_i \times V_j$, podem ser adjacentes ou não-adjacentes entre si. Como exemplo, citamos um dos problemas mais famosos que se insere neste contexto, o problema da k-coloração, onde deseja-se verificar se os vértices de um grafo podem ser particionados em k conjuntos independentes $V_1, \ldots, V_k$ (sem restrições externas). Sabe-se que esse problema é polinomial para $k \leq 2$ e *NP*-Completo para $k \geq 3$ [Bransdstädt (1996)].

Brandstädt propôs uma generalização dos grafos split, que definiu uma nova classe de grafos, a classe dos *grafos*-(k, ℓ), a qual também chamamos de grafos split generalizados, como sendo aquela formada pelos grafos cujo conjunto de vértices pode ser particionado em k conjuntos independentes e ℓ cliques. Brandstädt $(1996, 1998)$ considerou em particular as classes de grafos-$(2, 1)$, grafos-$(1, 2)$ e grafos-$(2, 2)$, apresentando algoritmos polinomiais para reconhecê-las. Feder *et al.* (1999) também apresentaram algoritmos polinomiais para o reconhecimento destas classes que surgiram como sub-produto de algoritmos de partição em subgrafos densos e esparsos. Por outro lado, sabe-se que reconhecer grafos-(k, ℓ) para $k \geq 3$ ou $\ell \geq 3$ é NP-Completo [Bransdstädt (1996)]. Como por exemplo, podemos considerar a classe dos grafos-$(k, 0)$, que corresponde ao problema de reconhecer se um dado grafo é k-colorível (seu reconhecimento é NP-Completo para $k \geq 3$).

Como o reconhecimento desta classe é NP-completo, para $k \geq 3$ ou $\ell \geq 3$, alguns autores estudaram

o problema quando restrito à subclasses de grafos. Hell *et.al* (2004) apresentaram uma caracterização por subgrafos proibidos e um algoritmo de reconhecimento com complexidade $O(n(n+m))$ para os grafos cordais-(k,ℓ). Bravo *et.al* (2011) apresentaram uma caracterização por subgrafos proibidos e um algoritmo de reconhecimento com complexidade $O(n+m)$ para os cografos-(k,ℓ).

Embora o reconhecimento de grafos-$(2,1)$ seja polinomial, pouco se sabe sobre as características da família das obstruções minimais de grafos-$(2,1)$. Sendo assim, o objetivo deste trabalho é demonstrar algumas propriedades inerentes das obstruções de grafos-$(2,1)$, e em seguida fazer uso destas para desenvolver um algoritmo eficiente de enumeração das obstruções minimais de grafos-$(2,1)$ com até nove vértices, facilmente extensível para grafos de ordem maior.

2. Preliminares

Dado um grafo simples $G = (V, E)$, denotamos por $\overline{G}$ o complemento de G. Para $V' \subseteq V$, denotamos por $G[V']$ o subgrafo de G induzido por V'. Uma *clique* (*conjunto independente*) é um subconjunto de vértices que induz um subgrafo completo (sem arestas), não necessariamente maximal e denotada por K_p (I_p) uma clique (conjunto independente) de p vértices.

Um grafo G é chamado grafo-$(2,1)$, ou simplesmente $(2,1)$, se seu conjunto de vértices pode ser particionado em 2 conjuntos independentes e uma clique. Tal classe de grafos é uma particularização da definição de grafo-(k,ℓ). Um grafo G é um *grafo*-(k,ℓ) se o conjunto de vértices V pode ser partiticionado em k conjuntos independentes e ℓ cliques

Uma obstrução de um grafo G é um subgrafo de G chamado de G' o qual impede que G tenha certa propriedade.

Um conjunto S é dito **minimal** em relação a uma determinada propriedade Π se S satisfaz Π, e não existe $S^{'} \subset S$ que satisfaça Π. Desta forma, uma obstrução minimal de um grafo-$(2,1)$ é qualquer grafo G minimal com relação a propriedade de *não ser um grafo*-$(2,1)$. Ou seja, G é um grafo que não é $(2,1)$ e qualquer subgrafo induzido de G é $(2,1)$.

Queremos com esse trabalho construir todas as obstruções para grafos-$(2,1)$, e focaremos em todos as obstruções minimais com até 9 vértices, e provaremos também que tais obstruções são minimais.

3. Compreensão Geral

Uma parte crucial para o desenvolvimento do algoritmo que será apresentado aqui é a observação de que toda obstrução minimal de um grafo-$(2,1)$ é necessariamente um grafo-$(2,2)$ e um grafo-$(3,1)$ simultaneamente. Um lema trivialmente extraído desse fato é o de que toda obstrução minimal de um grafo-$(2,1)$ contém dois conjuntos independentes, uma clique e um conjunto com um único vértice.

O teorema e sua prova são dados a seguir.

Teorema 1. *Toda obstrução minimal de um grafo-$(2,1)$ é necessariamente um grafo-$(2,2)$ e um grafo-$(3,1)$ simultaneamente, onde uma das partições é composta por um único vértice.*

Demonstração. Seja G uma obstrução minimal de grafo-$(2,1)$, isso significa que existe um vértice $v \in V(G)$ tal que a remoção desse vértice torna G é um grafo-$(2,1)$.

Suponha G' como um subgrafo induzido por $V(G) \setminus \{v\}$. Pela definição de minimal, podemos garantir que o grafo G' pode ser particionado em dois conjuntos independentes e uma clique, pois G é minimal. Portanto, $\{v\}$ é simultaneamente um conjunto independente e uma clique, logo G é um grafo-$(2,2)$ e G é um grafo-$(3,1)$. □

A partir desse teorema, tentamos atingir o objetivo de se construir uma obstrução minimal dos grafos-$(2,1)$ construindo primeiramente um grafo G' que é $(2,1)$ e em seguida, geramos o grafo G a partir da adição de um vértice especial v em G', e este deverá ser ligado a alguns vértices já existentes de forma a obter uma obstrução.

Notem que nossa proposta não é aplicar a ingênua abordagem de a partir de um dado n implementar um algoritmo exaustivo que enumera todos os grafos não isomorfos de tamanho máximo n, e em seguida testar quais destes são obstruções minimais. Em nossa abordagem poucas ligações (em especial as referentes ao último vértice) serão testadas por força bruta. Para tal, e com o objetivo de reduzir o tempo computacional e o número de grafos isomorfos a serem testados, faremos uso de características intrínsecas de grafos-$(2,1)$.

4. Características intrínsecas de um grafo-$(2,1)$

A prova do lema 1 nos mostra que obstruções de grafos-$(2,1)$ só existem a partir de 6 vértices, e também é possível mostrar que todo grafo-$(2,1)$ utilizado para gerar uma obstrução contém o subgrafo ($2K_2$) ilustrado na figura seguinte.

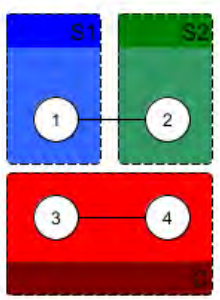


Figura 1: Subgrafo G' de qualquer grafo-$(2,1)$ utilizado. S_1, S_2 são conjuntos independentes e C é a clique de G'.

Lema 2. *Não existem obstruções dos grafos-$(2,1)$ com menos de 6 vértices.*

Demonstração. Pela dissertação de Silva (2011) [7] temos que um grafo G é (2,1) se e somente se o mesmo contiver uma clique C que intersecta todo ciclo ímpar de G. Portanto, basta demonstrar que não existem grafos com 5 ou menos vértices que possuam ciclos ímpares disjuntos.

Como o menor ciclo ímpar formado em um grafo simples sem laço é um ciclo de tamanho 3 (C_3), em um grafo com no máximo 5 vértices ao se formar um C_3 sobram apenas 2 vértices ainda disjuntos desse C_3, tornando assim impossível a construção de outro ciclo ímpar disjunto do primeiro. Logo, como C_3 é uma clique temos que não existem obstruções de grafos-$(2,1)$ com menos de 6 vértices. □

Lema 3. *Se G é uma obstrução minimal de um grafo-$(2,1)$ então:*

(a) O conjunto de vértices de G' pode ser particionado em 2 conjuntos independentes, S_1, S_2, e 1 clique, C, onde $|V(C)| \geq 2$ e existe uma aresta entre algum vértice de S_1 e algum vértice de S_2.

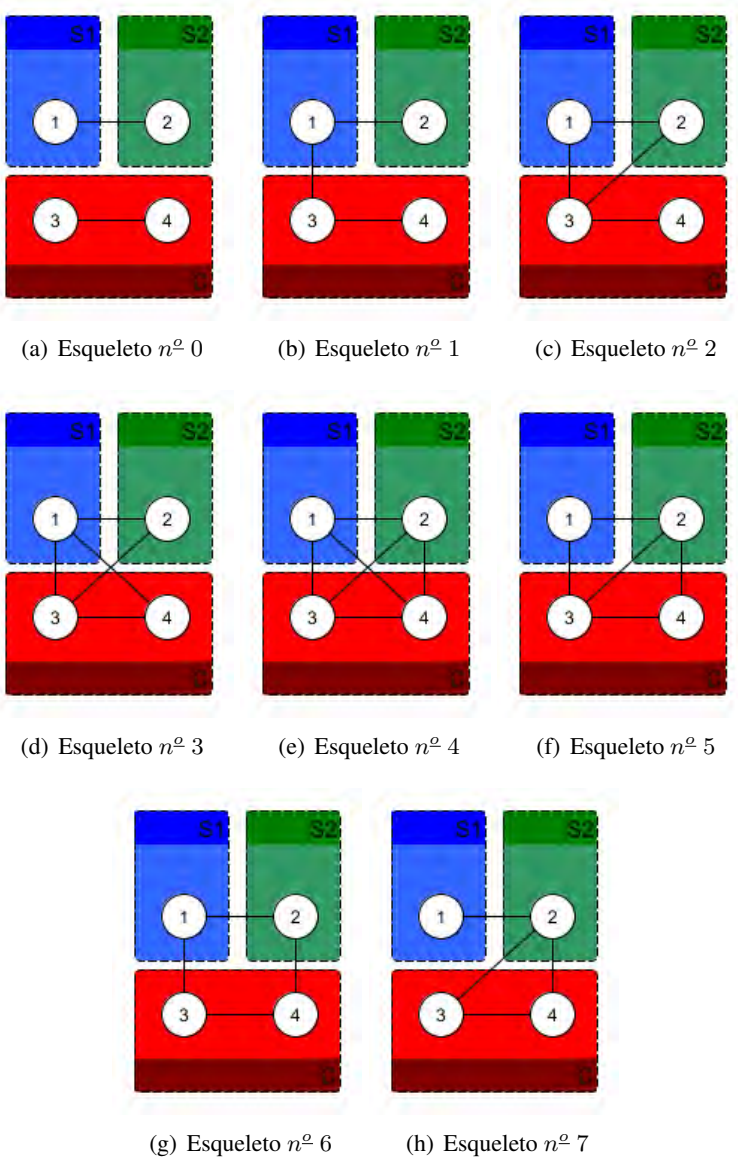
(b) Se G é uma obstrução minimal de grafos-$(2,1)$ então G não contém vértices de grau menor ou igual a 1.

Demonstração. (a) Basta observar, por absurdo, que se não existe aresta entre S_1 e S_2 então G' é um grafo-$(1,1)$ e portanto G é um grafo-$(2,1)$. Além disso, se $|C| = 1$ então:

- Se $v \in C$ possuir grau maior que 1 então podemos, sem perda de generalidade, inserir um vizinho de v em C. Absurdo, pois G não é um grafo-$(2,1)$.
- Senão, v possui grau zero e G é um grafo-$(2,1)$. Contradição.

(b) Seja G uma obstrução minimal e $w \in V(G)$ um vértice de grau menor ou igual a um. Por questão de minimalidade, $G[V \setminus \{w\}]$ é um grafo-$(2,1)$. Neste caso, w poderá ser adicionado ou a S_1 ou a S_2 o que implica em G ser $(2,1)$, uma contradição. □

Considerando os lemas apresentados acima, podemos agora construir nosso algoritmo. O algoritmo gera a partir do grafo $2K_2$, que sabemos que existirá em G', grafos bases que foram chamados de esqueletos internos, os quais são todos os subgrafos não isomorfos obtidos de todas as possíveis combinações de adição de arestas no subgrafo $2K_2$ inicial. A Figura 2 ilustra todos os sete possíveis esqueletos internos de G'.



(a) Esqueleto n^o 0 (b) Esqueleto n^o 1 (c) Esqueleto n^o 2

(d) Esqueleto n^o 3 (e) Esqueleto n^o 4 (f) Esqueleto n^o 5

(g) Esqueleto n^o 6 (h) Esqueleto n^o 7

Figura 2: Esqueletos de um grafo-$(2,1)$

4. Mapeamento dos vértices

Se desejamos construir uma obstrução com n vértices então ainda falta mapear os $n-5$ vértices restantes (os 4 vértices pertencentes ao esqueleto e o vértice especial para gerar a obstrução que já foram mapeados). Para isso, repetimos a idéia do esqueleto interno para os vértices que faltam sendo que dessa vez temos $\lceil (n-5)/4 \rceil$ grupos, cada um com até 4 vértices. Mais especificamente, para

cada grupo, fazemos um mapeamento entre os quatro vértices do grupo e os vértices dos possíveis grafos não isomorfos com o mesmo número de vértices. Um mapeamento entre um grupo e um grafo não isomorfo com 4 vértices, define um esqueleto que não tem uma estrutura definida e chamaremos de *externo*. Notem que quando $n \leq 9$ temos apenas um grupo. A partir de agora, analisaremos o caso em que $n = 9$, dado que este é nosso caso teste.

Na figura abaixo, temos uma possível configuração inicial para nosso algoritmo:

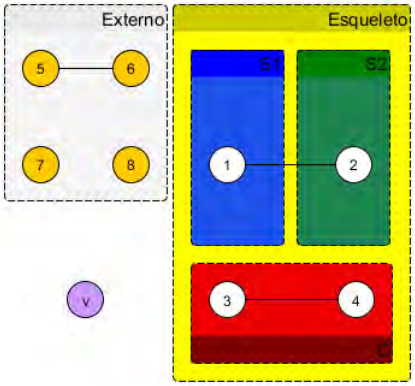


Figura 3: Possível configuração inicial para a geração de uma Obstrução dos grafos-$(2, 1)$

5. Obtenção da obstrução

Agora que temos todos os vértices mapeados, o próximo passo é adicionar os vértices de *Externo* ao esqueleto interno de todas as formas possíveis mantendo o esqueleto como um grafo-$(2, 1)$. Para isso, cada vértice de *Externo* é adicionado à uma das partições já existentes do grafo-(S_1, S_2, C), respeitando a estrutura do Externo e as regras da partição, onde o mesmo será incluído. Por exemplo: se um vértice $u \in$ *Externo(V)* tenta ser incluído em um dos conjuntos independentes (S_1, S_2), porém um de seus vizinhos em Externo já está incluído no mesmo conjunto independente, então não é permitido que u faça parte desse conjunto, pois assim ele deixaria de ser um conjunto independente.

Como sabemos, o número mínimo de vértices que uma obstrução minimal dos grafos-$(2, 1)$ pode ter é 6. Portanto para fins didáticos, o exemplo da obtenção de uma obstrução será feita com 6 vértices.

Assim temos como configuração inicial:

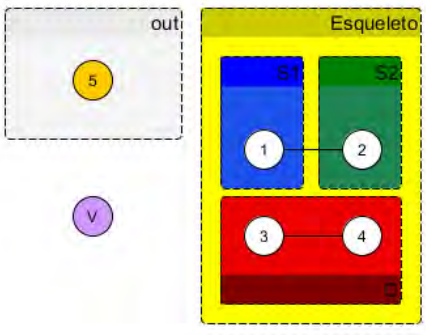
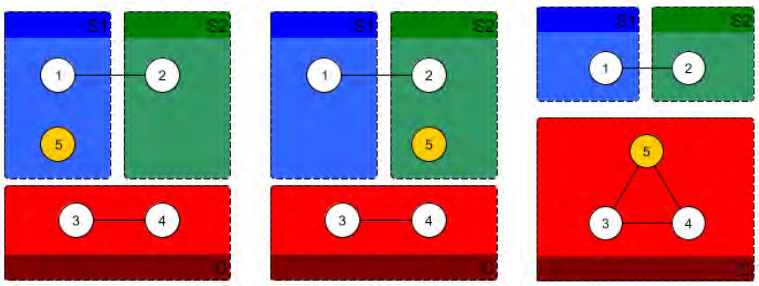


Figura 4: Configuração Inicial para exemplo

As possíveis posições do vértice de *Externo* no esqueleto são:



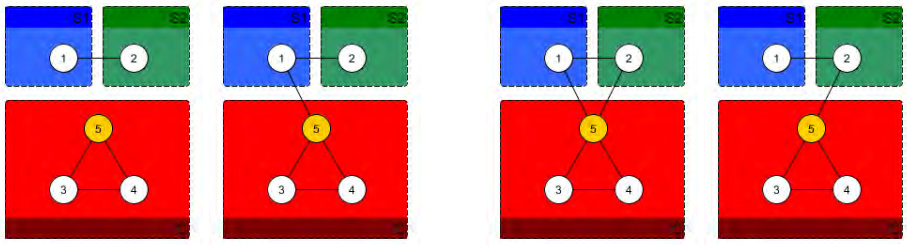
(a) Novo esqueleto n^o 0 (b) Novo esqueleto n^o 1 (c) Novo esqueleto n^o 2

Figura 5: Os possíveis esqueletos com a adição do vértice de *Externo*

Agora, usaremos o item c da Figura 5 (*novo esqueleto* n^o 2) como continuação do nosso exemplo.

O próximo passo nesse algoritmo é a de enumerar todas as possibilidades de adição de arestas dos vértices $u \in V(Externo)$ para os outros vértices do esqueleto interno sem quebrar a característica da partição onde ele está inserido. (Exemplo: um vértice $u \in S_1$ não pode adicionar uma aresta para um vértice $w \in S_1$), para esses grafos daremos o nome de *compostos*.

Usando o novo esqueleto n^o 2 como exemplo temos:



(a) Composto #0 (b) Composto #1 (c) Composto #2 (d) Composto #3

Figura 6: Os possíveis compostos com a adição das arestas

Vamos olhar para o item a da Figura 6 (Composto #0) para o próximo passo.

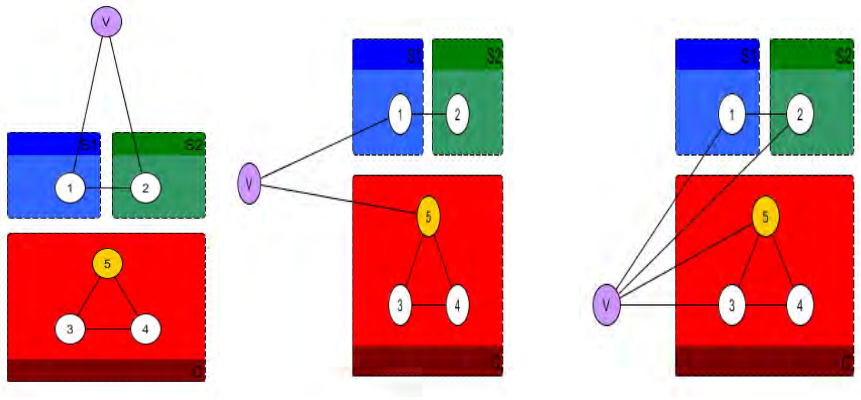
Nos compostos resultantes verificamos que se algum vértice possui grau igual a zero, em caso positivo, o removemos. E, se após remoções o esqueleto composto encontra-se com menos de seis vértices então o descartamos. Tal operação sempre poderá ser aplicada devido ao item b do Lema 3.

Pelo teorema acima, podemos garantir que um composto G' não pode ter vértices de grau zero, pois o grafo G resultante terá vértices de grau menor ou igual a zero.

A partir dos compostos obtidos acima adicionamos o vértice v e enumeramos todas as possíveis combinações de adições de arestas de v ao composto. Os grafos gerados por esse passo foram chamados de *candidatos*. Abaixo são mostrados alguns dos candidatos gerados para o Composto #0.

Com todos os candidatos para grafos com n vértices gerados, aplicamos o algoritmo de Brandstädt (1996, 1998)[1, 2] em cada um deles obtendo aqueles que são obstruções, e em seguida testamos a minimalidade. No nosso exemplo, o candidato #0 é uma obstrução.

O uso desse algoritmo para a construção de obstruções com n vértices, apesar de não ser polinomial, é uma boa abordagem, pois além de filtrar diversos grafos isomorfos por reduzir a quantidade possível de grafos, também garante que os grafos gerados são todos candidatos a obstrução, pois antes do estágio de candidato, todo grafo é garantidamente $(2, 1)$. Desta forma, grafos desnecessários, como grafos contendo subgrafos proibidos, nunca serão gerados na construção de G'.



(a) Candidato #0 (b) Candidato #1 (c) Candidato #2

Figura 7: Alguns candidatos gerados pelo algoritmo

6. Corretude do algoritmo

Teorema 4. *O algoritmo está correto, isto é, gera todas as obstruções minimais.*

Demonstração. Pelo Teorema 1, sabemos que todas as obstruções minimais de grafos-$(2,1)$ são simultaneamente um grafo-$(2,2)$ e um grafo-$(3,1)$ onde uma das partições contém apenas um vértice. Seja v esse vértice de G. Sabemos também que $G' = G[V \setminus \{v\}]$ é um grafo-$(2,1)$.

Nosso algoritmo se baseia em, primeiramente, enumerar todos os possíveis grafos G', e para cada um deles adicionarmos o vértice v e para cada possível combinação de arestas de v para vértices pertencentes a $V(G')$, testarmos se o grafo resultante é uma obstrução, portanto para a corretude do algoritmo basta apenas mostrarmos que todos os grafos G' foram contemplados pelo nosso algoritmo.

Os esqueletos internos $0, \ldots, 7$ são todos os possíveis subgrafos não isomorfos induzidos pelos quatro vértices do subgrafo $2K_2$ inicial.

Como toda obstrução contém como subgrafo induzido um desses esqueletos, nosso algoritmo tem como propriedade chave enumerar todas as obstruções que podem ser obtidas a partir da fixação de cada um desses esqueletos.

Neste ponto, falta analisar os vértices restantes de $G'[V(G') \setminus \{1,2,3,4\}]$ que possui 4 vértices, nos dando 12 possíveis grafos não isomorfos que denominamos externos.

Combinando os 12 possíveis grafos com os 8 esqueletos e as combinações de arestas entre eles obtemos todos os grafos G' possíveis. Note que, se algum vértice fica isolado o mesmo é removido antes da inserção de v e portanto as obstruções com $6, 7$ e 8 vértices também são geradas. □

7. Considerações Finais

Neste trabalho demonstramos algumas propriedades das obstruções de grafos-$(2,1)$, e em seguida fizemos uso destas para obter um algoritmo competitivo de enumeração das obstruções minimais de grafos-$(2,1)$ com até nove vértices, facilmente extensível para grafos de ordem maior.

Para ilustrar quão úteis e promissoras são as etapas definidas para o nosso algoritmo, primeiramente observe que existem 2^{36} grafos distintos com até 9 vértices (vértices rotulados), e portanto a enumeração de cada um deles seguida de uma posterior filtragem de casos isomorfos e uma verificação de obstrução minimal de $(2,1)$, certamente não nos fornece um método executável na prática.

Nossa abordagem visa portanto eliminar uma grande gama de casos isomorfos em nossa análise. Isso é possível explorando decisões que podem ser facilmente tomadas, uma vez que propriedades intrínsecas das obstruções de grafos-$(2,1)$ são conhecidas.

Para ilustrar quão impactante são as decisões tomadas no projeto de nosso algoritmo, em um primeiro teste para obstruções com até nove vértices, foi efetuada uma execução do algoritmo de enumeração sem nenhum tratamento para isomorfismo ou duplicatas (repetições de um mesmo grafo) entre os indivíduos finais gerados. Além disso, não foi considerado a remoção de vértices de grau zero nos candidatos e usamos esqueletos externos de tamanho entre um e quatro vertices. Neste caso, foram gerados 8gb de dados em um espaço de 6h. Notem que, embora um grande número de indivíduos tenham sido gerados, o algoritmo já apresentou uma execução viável.

Em uma segunda empreitada, removendo duplicadas (indivíduos indênticos), usando esqueletos externos de tamanho exatamente quatro, e eliminando vértices de grau zero nos candidatos, foram gerados apenas 22mil grafos em 94min sumarizando 8mb de dados. Uma melhoria considerável, como pode ser visto.

Por fim, uma terceira geração foi obtida a partir da elinação de casos isomorfos dentre os indivíduos da segunda geração resultando em 1026 obstruções minimais de grafos-$(2,1)$ em 80min num arquivo final de 500Kb. Sendo 174min o tempo de execução de todo o processo necessário para produção da terceira geração.

Toda implementação dos algoritmos foi feita na Linguagem Java, utilizando o framework JGraphT e em um segundo momento em javascript com o framework vis.js para a geração da visualização do resultado

Para eliminação dos casos isomorfos implementamos o algoritmo VF2 de Cordella et al (2004). [4] disponibilizado pelo framework JGraphT

Em nossos experimentos foi utilizado uma máquina Lenovo e431, com intel core i5 e 4gb de memória RAM.

O arquivo gerado com as obstruções pode ser acessado pelo link abaixo.

http://www2.ic.uff.br/~ueverton/obstrucao21/Apendice.pdf

Referências

[1] **Bransdstädt, A.** Partitions of graphs into one or two independent sets and cliques. *Discrete Mathematics* 152 (1996) 47 – 54.

[2] **Bransdstädt, A.** The complexity of some problems related to graph 3-colorability. *Discrete Applied Mathematics* 89 (1998) 59 – 73.

[3] **Bravo, R. S. F., Klein, S., Nogueira, L. T., and Protti, Fábio** Characterization and recognition of P_4-sparse graphs partitionable into independent sets and cliques. *Discrete Applied Mathematics* 159 (2011) 165 – 173.

[4] **Cordella, Luigi P., et al.** A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.10 (2004) 1367–1372.

[5] **Golumbic, M. C.** *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[6] **Hell, P., Klein, S., Nogueira, L. T., and Protti, F.** Partitioning chordal graphs into independent sets and cliques. *Discrete Applied Mathematics* 141 (2004) 185 – 194.

[7] **Silva, J. S.** Obstruções minimais de grafos-$(2,1)$. *Dissertação de mestrado*, Universidade Federal Fluminense (2011)