GRAPH FILTER: UM SOFTWARE INTUITIVO PARA FILTRAGEM DE GRAFOS COM PROPRIEDADES CUSTOMIZADAS PELO USUÁRIO.

Denilson Paula de Oliveira Ribeiro[1], Átila Arueira Jones[2]

[1] *Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais; Campus Juiz de Fora, denilsonrib9@gmail.com*
[2] *Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais; Campus Juiz de Fora, atila.jones@ifsudestemg.edu.br*

RESUMO: Este trabalho apresenta o software *Graph Filter (license GPL v3.0)*, cuja finalidade é filtrar uma lista de grafos seguindo propriedades numéricas estabelecidas e customizadas pelo usuário, permitindo ainda que este insira equações (ou inequações) que envolvam diversos invariantes de grafos. O uso do programa não exige conhecimento prévio do usuário em nenhuma linguagem de programação, pois este conta com interface intuitiva e funcionalidades de exportação visual do grafo. O software tem como objetivo principal auxiliar o pesquisador para estabelecer ou refutar conjecturas de forma rápida e simples. Este foi desenvolvido em linguagem C#, com uso de pacote de interpretador de equações e visualização do grafo. O *Graph Filter* está publicado em página própria e pode ser instalado em qualquer sistema Windows.

Palavras-chave: Grafo; *Graph Filter*; Filtragem; Algoritmo;

1. INTRODUÇÃO

Um grafo é um par (V, E), onde V é o conjunto de vértices e E o conjunto das arestas que ligam pares de vértices. Em uma pesquisa científica em Teoria dos Grafos, bem como outras áreas, é comum a análise de diversos exemplos, seja com a finalidade de buscar padrões para elaboração de novos resultados ou ainda para refutar conjecturas através de construção de contraexemplos. Porém observar um grafo com muitos vértices começa a se tornar inviável pela dificuldade tanto de visualizar quanto pelo estudo da propriedade em si. Claramente este processo é mais simples quando trabalhamos computacionalmente, porém ainda há a dificuldade em analisar grandes quantidades de grafos ao mesmo tempo, pois requer conhecimentos de programação por parte do pesquisador para implementar os grafos e, principalmente, o cálculo dos invariantes que deseja observar na estrutura. Neste projeto foi desenvolvido o software *Graph Filter*, licenciada em GNU GPL v3.0, com ferramentas tanto da Teoria dos Grafos, quanto para a Teoria Espectral de Grafos, cuja principal finalidade é realizar a filtragem e visualização de grafos e então disponibilizá-los para estudos futuros, auxiliando pesquisadores teóricos da área ao estabelecer, verificar ou refutar conjecturas através de experimentos com grafos.

ISBN: 978-65-5623-103-7 | http://editora.pucrs.br

Existem alguns softwares que possuem finalidade semelhantes ao *Graph Filter*, valendo assim citá-los e destacar as diferenças. São estes:

- AutoGraphiX - AGX-III (CAPOROSSI, 2017): O seu principal objetivo é pesquisar grafos extremais, ou seja, grafos que minimizem ou maximizem um invariante do grafo (ou uma função de invariantes do grafo). O software *Graph Filter* se diferencia do AGX-III principalmente pelo fato do usuário poder inserir equações ou inequações que desejar, podendo esta envolver diversos invariantes em uma mesma expressão.
- Sage (in Graph Theory) (THE SAGE DEVELOPERS, 2020): trata-se de um software de matemática que possui diversos recursos, inclusive em Teoria dos Grafos. Permitindo, em particular, que o usuário implemente um script na linguagem do software para efetuar uma filtragem de uma lista de grafos. Portanto trata-se de um programa exige conhecimentos básicos de programação por parte do usuário. O *Graph Filter* se diferencia principalmente neste ponto, pois propõe justamente apresentar um layout intuitivo e que não exige conhecimento, do usuário, em código de programação para efetuar a filtragem, incorporado ainda a uma ferramenta de visualização e exportação dos grafos obtidos.

No decorrer das próximas seções apresentaremos detalhes do funcionamento e aplicação do software *Graph Filter*, desenvolvido neste trabalho.

2. METODOLOGIA

O software *Graph Filter* foi implementado em C# utilizando o Microsoft Visual Studio versão Community 2019. Durante o seu desenvolvimento foram utilizados pacotes públicos com diferentes finalidades: Pacote GraphX para integração da ferramenta de visualização de grafos no software (SMIRNOV, 2016), Pacote Flee (PARLAK, 2017) que interpreta texto como uma equação matemática na linguagem; Pacote MetroFramework - Modern UI for WinForms (MAGNO, 2014) para a implementação da interface visual; Pacote GraphPlanarityTesting (NEPOŽITEK, 2019) para a adição da ferramenta de teste de planaridade em grafos; Pacote MaximumFlowProblem (RABIEC, 2017) para o cálculo de conectividade de arestas.

Todo o código está publicado para leitura na plataforma GitHub através do repositório **github.com/GraphFilter/GraphFilter** vinculado ao Zenodo com **doi.org/10.5281/zenodo.4047104**. No momento da submissão deste trabalho o software está na versão 1.1, como pode ser encontrado no release do repositório.

ISBN: 978-65-5623-103-7 | http://editora.pucrs.br

3. RESULTADOS E DISCUSSÕES

O software foi publicado e disponibilizado para a comunidade através de uma página própria, hospedada no Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais (IF Sudeste MG). Este site disponibiliza o arquivo do instalador do software, manual de usuário, autoria, links de páginas que disponibilizam arquivos em graph6 e acesso a visualização do código fonte via *GitHub*. A página pode ser acessada por **sistemas.jf.ifsudestemg.edu.br/graphfilter.** Detalharemos agora o funcionamento do software.

3.1. Funcionamento do Software

Em resumo, o funcionamento do *Graph Filter* se dá através da entrada da lista de grafos e condições impostas pelo usuário, estas são inseridas por equações digitadas e/ou *Check Box* que representam condições de verdadeiro ou falso. Ao iniciar a busca, o programa analisará cada grafo da lista, verificando se este satisfaz as condições impostas pelo usuário. Em caso afirmativo, o algoritmo armazena este grafo no arquivo de saída; em caso contrário, o grafo é descartado. Ao fim do processo o usuário terá uma nova lista de grafos, que são os filtrados da lista de entrada e que satisfazem as condições impostas. A Figura 1 representa um esquema do algoritmo.

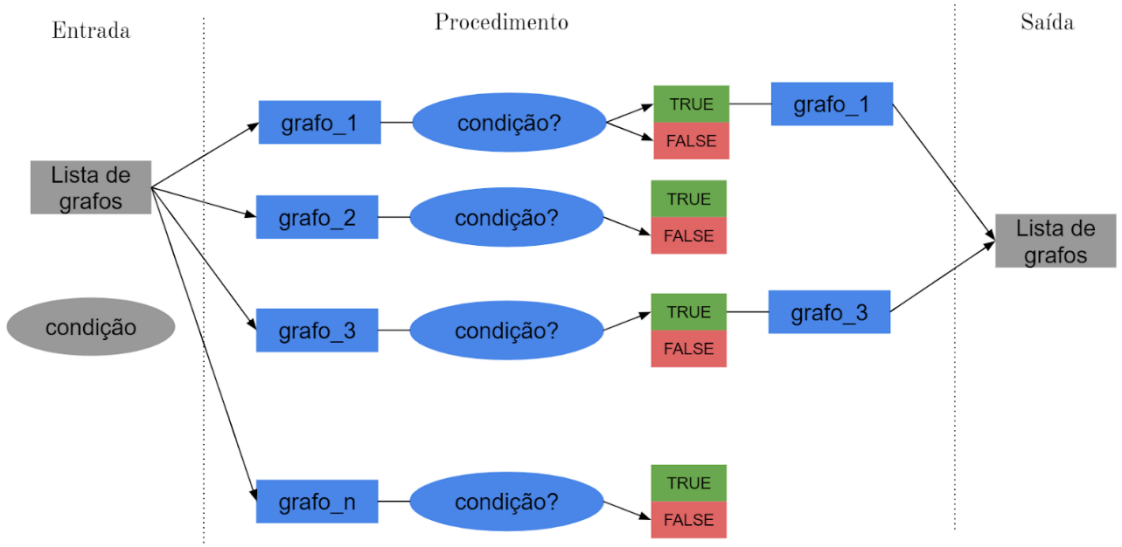

Figura 1. Esquema de Funcionamento do Software.

Fonte: Figura do autor.

ISBN: 978-65-5623-103-7 | http://editora.pucrs.br

3.1.1. Entrada do Software

O arquivo de entrada trata-se de um arquivo de texto contendo uma lista de grafos codificados no formato Graph6 (MCKAY, 2007) que, a título de curiosidade, trata-se de uma forma de representar um grafo com baixíssimo uso de memória computacional, pois é basicamente a transformação da matriz de adjacência em código binário e posteriormente conversão para a tabela ASCII, reduzindo um grafo a um conjunto de caracteres. Vale ressaltar que não há necessidade de o usuário ter conhecimento prévio sobre o Graph6, várias classes de grafos já estão codificadas neste formato em bibliotecas publicadas como o House of Graphs (BRINKMANN et al., 2013), que concentra diversos trabalhos sobre geração de grafos em classes, e disponibiliza para download esse acervo de grafos em formato Graph6, já o software *Nauty and Traces* (MCKAY; PIPERNO, 2014) possui um pacote que possibilita a geração de grafos pequenos incluindo algumas restrições simples, inclusive na seção *Graphs by Brendan McKay* disponibiliza para download uma lista de grafos já gerados neste formato.

3.1.2. Processamento

Para processar a lista de grafos, o software verifica quais as condições booleanas foram checadas e quais foram os invariantes definidos pelas equações (ou inequações) criadas pelo usuário. Na Tabela 1 são listados os invariantes booleanos (do tipo Verdadeiro ou Falso) já implementados no software.

O usuário pode ainda estabelecer quantas equações ou inequações desejar, que podem envolver diversos invariantes numéricos, utilizando as operações básicas da matemática. A atribuição de mais de uma equação ou inequação é feita através de conectivos lógicos “AND” ou “OR” inseridos pelo usuário. Os invariantes numéricos já implementados estão listados na Tabela 2.

ISBN: 978-65-5623-103-7 | http://editora.pucrs.br

Tabela 1. Invariantes Booleanos.

Regular	Maior Autovalor de adjacência é inteiro	Integral
Regular com grau k	Maior Autovalor laplaciano é inteiro	L-Integral
Hamiltoniano	Maior Autovalor da laplaciana sem sinal é inteiro	Q-Integral
Planar	Algum Autovalor da adjacência é inteiro	Acíclico
Algum Autovalor da laplaciana é inteiro	Algum Autovalor da laplaciana sem sinal é inteiro	Conexo

Tabela 2 - Invariantes Numéricos.

INVARIANT	**CÓDIGO**	**INVARIANT**	**CÓDIGO**
Maior (e segundo) autovalor	lambda1, lambda2	Número de árvores geradoras	spnt
Energia (Adjacência)	Ea	Nulidade	nul
Maior (e segundo) autovalor Laplaciano	mu1, mu2	Conectividade algébrica	ac
Energia Laplaciana	El	Nº de arestas	m
Diâmetro	diam	Número clique	omega
Grau máximo, mínimo, médio	Delta, d, delta	Cardinalidade do emparelhamento máximo	M
Número independente	Nc	Cintura	g
Conectividade de arestas	ec	N° de componentes	Nc
Ordem	n	Número Cromático	chi

Ao inserir o código do invariante na equação, o usuário deve ainda inserir "*_G"* se deseja que o invariante seja calculado no próprio grafo que consta na lista do arquivo de entrada, ou inserir "*_cG"* para o cálculo no complemento do grafo referente ao que consta na lista, ou ainda inserir "*_lG"* para o cálculo no grafo linha. Por exemplo: "*diam_cG*" representa o diâmetro no grafo complementar de G. A Figura 2 representa uma captura

ISBN: 978-65-5623-103-7 | http://editora.pucrs.br

de tela do software exibindo o arquivo dado como entrada e as condições impostas pelo usuário, a saber, hamiltoniano e satisfaz a equação $n(G) + \alpha(\overline{G}) \geq 5 \cdot ac(\ell(G))$.

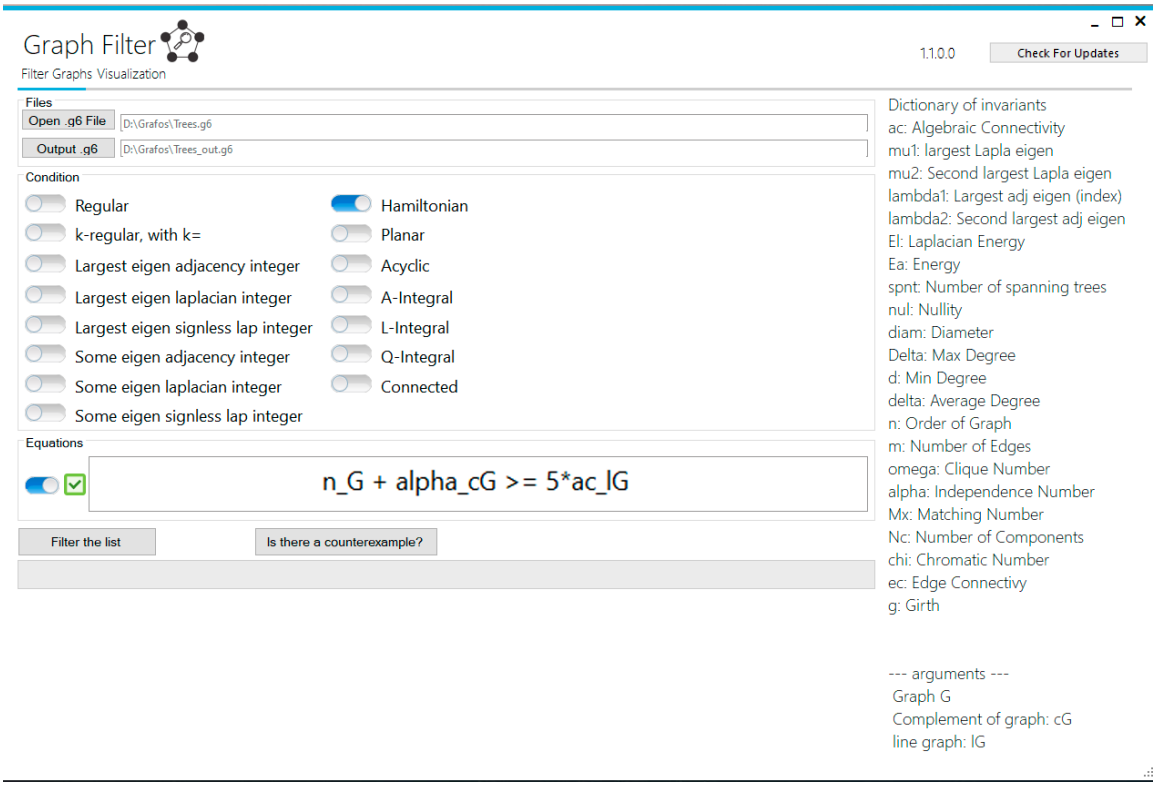

Figura 2. Captura de tela da aba "Filter Graphs" do Graph Filter.

Fonte: Figura do autor.

3.1.3. Saída

Existem duas formas de retorno para o usuário, representadas pelos botões "*Filter the List*" e "*Is there a counterexample*".

O primeiro executa a filtragem dos grafos da lista do arquivo de entrada, retornando todos os grafos que satisfazem as condições impostas pelo usuário em um arquivo de formato g6 (este é criado pelo usuário anteriormente utilizando o botão "*Output*"). Após a filtragem completa, o programa exibe um relatório informando a proporção de sucesso da filtragem. Esta opção é útil para visualizar padrões estruturais ou numéricos em uma lista de grafos, o que auxilia o pesquisador a compreender conceitos, elaborar novos resultados e constatar conjecturas de forma experimental.

Enquanto o botão "*Is there a counterexample*" busca dentre a lista de grafos dado na entrada algum grafo que não satisfaça as condições impostas pelo usuário, ou seja, retorna um grafo contraexemplo, caso exista, assim que for encontrado. Recurso útil para

ISBN: 978-65-5623-103-7 | http://editora.pucrs.br

refutar conjecturas, pois não necessita de fazer uma filtragem completa na lista de grafos de entrada, parando a busca assim que for encontrado o primeiro contraexemplo.

3.1.4. Visualização e Exportação

O software também tem a funcionalidade de visualização e exportação de grafos, conforme visto na Figura 3.

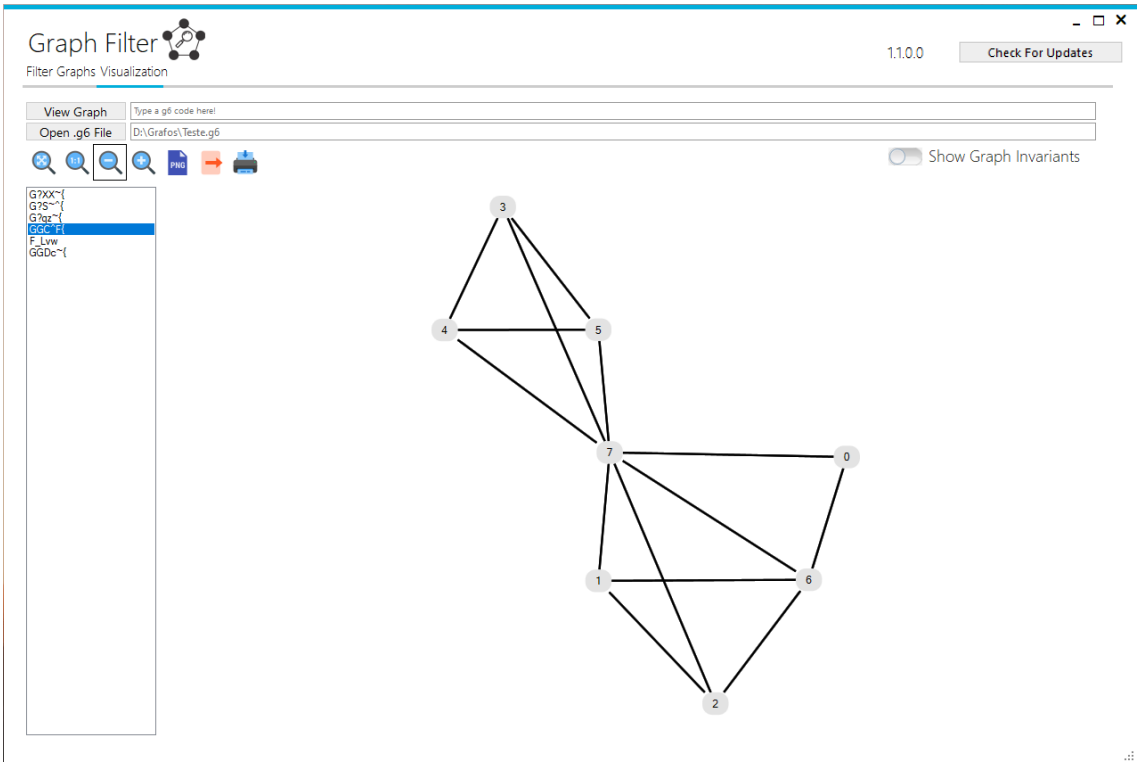

Figura 3. Captura de tela da aba "*Visualization*" do *Graph Filter*.

Fonte: Figura do autor.

A função principal desta aba é proporcionar a visualização da lista de grafos que foi filtrada na aba anterior, possibilitando também a inserção manual de um grafo em formato g6 apenas escrevendo-o na caixa de texto ao lado do botão "*View Graph*". Ao inserir um arquivo com grafos em formato g6 através do "*Open .g6 File*", os grafos serão inseridos na lista, para visualizar qualquer um deles basta selecionar o grafo escolhido. O software ainda permite exportar para PNG, TikZ ou PDF.

No canto direito o botão "*Show Graph Invariants*" exibe uma tabela com todos os invariantes numéricos, que já existem no programa, calculados para o grafo que está sendo visualizado.

ISBN: 978-65-5623-103-7 | http://editora.pucrs.br

4. CONCLUSÕES

Entendemos que esta é uma ferramenta útil para pesquisadores da área de Teoria dos Grafos, pois trata-se de um software capaz de filtrar grandes quantidades de grafos com condições impostas e equações totalmente customizáveis pelo usuário, ressaltando que não há necessidade de conhecimento em programação por parte do usuário, pois apresenta uma interface intuitiva e de fácil utilização. Além da disponibilização do software para download com instalador intuitivo por meio de um website com guia de usuário de fácil leitura e entendimento. O programa pode ser instalado e executado em qualquer sistema Windows, mas está sendo trabalhado para que em breve tenha compatibilidade com outros sistemas operacionais. Naturalmente o trabalho continua em evolução através da atualização do software para novas versões que trarão a implementação de novos invariantes e novas funcionalidades.

REFERÊNCIAS

BRINKMANN, G. et al. House of Graphs: A database of interesting graphs. *Discrete Applied Mathematics*, v. 161, n. 1–2, p. 311–314, 2013.

CAPOROSSI, G. Variable Neighborhood Search for extremal vertices : The AutoGraphiX-III system. *Computers and Operations Research*, v. 78, p. 431–438, 1 fev. 2017.

MAGNO, D. **Metro Modern UI.** Disponível em: <https://www.nuget.org/packages/MetroModernUI/1.4.0>. Acesso em: 16 set. 2020.

MCKAY, B. D. graph6 and sparse6 graph formats. *Computer Science Department, Australian National University*, September, v. 12, p. 2007, 2007.

MCKAY, B. D.; PIPERNO, A. Practical graph isomorphism, {II}. *Journal of Symbolic Computation*, v. 60, n. 0, p. 94–112, 2014.

NEPOŽITEK, O. **GraphPlanarityTesting.** Implementation of the Boyer-Myrvold algorithm for planarity testing of graphs. Disponível em: <https://github.com/OndrejNepozitek/GraphPlanarityTesting>. Acesso em: 16 set. 2020.

PARLAK, **M. Flee** (Supports .Net Core 2.0). Fast Lightweight Expression Evaluator. Disponível em: <https://github.com/mparlak/Flee>. Acesso em: 16 set. 2020.

RABIEC, L. **MaximumFlowProblem**. Ford-Fulkerson algorithm for Maximum Flow Problem. Disponível em: <https://github.com/lucasrabiec/MaximumFlowProblem>. Acesso em: 16 set. 2020.

SMIRNOV, A. **GraphX for .NET**. Disponível em: <https://github.com/panthernet/GraphX>. Acesso em: 16 set. 2020.

THE SAGE DEVELOPERS. **SageMath**, the Sage Mathematics Software System, 2020.

ISBN: 978-65-5623-103-7 | http://editora.pucrs.br