

Graph Filter 2.0: nova versão do software para filtragem de grafos.

Lavínia Beghini de Castro

IF Sudeste MG
Juiz de Fora – MG
lavinia@gmail.com

Átila Arueira Jones

IF Sudeste MG
Juiz de Fora - MG
atila.jones@ifsudestemg.edu.br

RESUMO

O presente artigo tem o objetivo de demonstrar os resultados e a evolução do desenvolvimento do software *Graph Filter* da antiga versão 1.0 para sua nova versão 2.0. Este programa tem a finalidade de filtrar uma lista de grafos a partir de equações, inequações ou condições estabelecidas pelo usuário, de forma interativa e intuitiva. Nesta nova versão o software conta com mais funcionalidades, compatibilidade com outros sistemas, exportação dos grafos filtrados em diversos formatos e melhor estruturação do código, possibilitando o crescimento do programa. O objetivo do trabalho é filtrar grafos visando auxiliar pesquisadores da área de Teoria de Grafos a observarem padrões, estabelecerem ou refutarem conjecturas. Apresentaremos aqui detalhes desta nova versão, sua motivação, seu desenvolvimento e principais funcionalidades.

PALAVRAS CHAVE. Grafo, Filtragem, Algoritmo.

**Tópicos: Teoria e Algoritmos em Grafos (TAG),
Otimização Combinatória (OC)**

ABSTRACT

This article aims to demonstrate the results and evolution of the development of the Graph Filter software from the previous version 1.0 to its new version 2.0. This program has the purpose of filtering a list of graphs from equations, inequalities or conditions established by the user, in an interactive and intuitive way. In this new version, the software has more features, compatibility with other systems, export of filtered graphs in different formats and better code structuring, enabling the program to grow. The objective of the work is to filter graphs aiming to help researchers in the field of Graph Theory to observe patterns, establish or refute conjectures. Here we will present details of this new version, its motivation, its development and main features.

KEYWORDS. Graph, Filtering, Algorithm.

**Paper topics: Theory and Algorithms in Graphs (TAG),
Combinatory Optimization (OC)**

1. Introdução

Um grafo é um par de conjuntos (V, E) , onde V é um conjunto de vértices e E conjunto de arestas formadas por pares de vértices. Pesquisadores de Teoria dos grafos analisam estas estruturas buscando padrões para elaboração de novos resultados, ou então refutar conjecturas através de contraexemplos. Porém, ao lidar com um grafo com muitos vértices e arestas, se torna inviável tanto para visualizar quanto para estudar alguma propriedade. Este processo é facilitado se o pesquisador tiver acesso a um software com recursos que auxiliem sua pesquisa, com layout objetivo que não exija nenhum conhecimento prévio em uma linguagem de programação por parte do usuário. Com a finalidade de auxiliar os estudos destes pesquisadores, foi desenvolvido o software *Graph Filter v1.0* publicado em [Ribeiro e Jones 2020]. Dando prosseguimento a isso, este trabalho visa aprimorar o *Graph Filter* para a versão 2.0, focando em expandir a compatibilidade para novos Sistemas Operacionais, aumentar o número de funcionalidades e melhorar a interface do software de forma a facilitar sua utilização. Descreveremos neste trabalho a motivação para mudança da linguagem de programação, pesquisa sobre as bibliotecas na sua implementação, o desenvolvimento do projeto em si e, por fim, o detalhamento do uso do software e suas funcionalidades.

2. Motivação

Interfaces de usuário são uma abstração do código que é executado pelos programas, o que tende a ser mais amigável para pessoas que não possuem domínio em linguagens de programação, tornando o software mais atraente e de melhor usabilidade. Baseando-se nessa premissa, foi desenvolvido o *Graph Filter*. É conhecida a existência de programas capazes de esboçar grafos, fazer varredura e calcular invariantes, como o *AutoGraphiX* e *Sage*. Porém estes não cumprem o mesmo propósito do *Graph Filter*, cuja comparação foi detalhada no artigo [Ribeiro e Jones 2020].

A primeira versão do software, obteve sucesso em seu objetivo. Tendo sido construída na linguagem de programação C#, com a interface feita com o auxílio da IDE (*Integrated Development Environment*) Microsoft Visual Studio, o programa contém, em resumo, as seguintes funcionalidades:

- Permite inserir uma equação para a filtragem;
- Permite selecionar condições específicas para a filtragem;
- Exibe opção de filtragem e de encontrar um contraexemplo;
- Filtra grafos a partir de um arquivo *graph6* para outro arquivo *graph6* [McKay 2007];
- Mostra uma visualização interativa dos grafos filtrados;
- Mostra mais informações sobre os grafos filtrados;
- Exporta os grafos para os formatos PNG e TikZ.

Muito embora bem encaminhado, haviam alguns impasses que afetavam seu desenvolvimento, tais como: demora na filtragem; implementação de novos invariantes; a falta de bibliotecas auxiliares específicas para grafos na linguagem de programação inicialmente escolhida; e a dificuldade de se criar executáveis do programa para outros Sistemas Operacionais que não o Microsoft Windows, já que a linguagem de programação C# é nativa desse sistema.

Dando continuidade a isso, a segunda versão resolve os obstáculos encontrados no ponto de base anterior, e, juntamente a isso, melhorar as funcionalidades de forma a trazer ainda mais facilidades nos intentos da proposta.

3. Pesquisa para evolução da versão

Inicialmente foram feitos estudos para traçar um método de resolver os problemas observados na primeira versão. O foco principal foi tentar inserir mais bibliotecas no projeto. A linguagem de programação Python é uma das linguagens mais utilizadas atualmente, como pode ser visto em [Zapponi 2020], e contém bibliotecas como a *NetworkX* [Aric et al. 2008], que possui várias funções de cálculos específicos para grafos, o que já possibilita um aumento significativo de

invariantes implementados. Mesmo que as funções pudessem ser implementadas do zero, não é vantajoso tentar competir com a quantidade de funcionalidades de uma biblioteca completa, pois sua reimplementação demoraria muito tempo e mudaria o foco do trabalho. Portanto, o objetivo inicial foi tentar fazer com que o C# conseguisse executar as funções desse pacote. Para tal, algumas opções foram levantadas e analisadas:

- **IronPython**

IronPython é uma implementação da linguagem Python integrada ao .NET, que é um *framework* da Microsoft que roda principalmente no Windows, e que possui grande integração com o C#. Com o IronPython seria possível reutilizar o código já pronto feito em C, porém, por ser uma implementação da linguagem Python e não a linguagem em si, não havia certeza se as bibliotecas em Python existiam para o IronPython. Além disso, as bibliotecas não poderiam utilizar a linguagem de programação C em sua base, pois o C# não possui compatibilidade.

- **Python.NET**

Assim como o IronPython, o Python.NET também fornece uma integração com o .NET e o C#, porém não implementa a linguagem Python. Devido a isso, não representa nenhuma vantagem para o projeto pois demoraria muito tempo para descobrir como importar a interface da versão 1.0, e caso não houvesse solução para a importação, seria necessário a reimplementação completa do projeto.

- **Python como um serviço**

Uma outra possibilidade seria chamar os *scripts* de Python direto do C#. Fazendo isso, não existiria problemas em reutilizar o que já estava pronto, mas com a desvantagem de que o usuário teria de instalar previamente o Python em sua máquina. Isso tiraria a autossuficiência do programa, além de que não é um processo otimizado chamar *scripts* para cada uma das funções a serem desenvolvidas.

- **Python**

A última alternativa considerada foi migrar toda a aplicação para a linguagem Python. Foi a última a ser considerada pois não aproveitaria a interface já feita em C#, e teria o tempo gasto com a reimplementação. Muito embora essas desvantagens fossem significativas, as vantagens eram numerosas: as bibliotecas de cálculos matemáticos eram garantidas em questão de funcionamento e disponibilidade; o código ficaria mais limpo, sem camadas de implementação de uma linguagem para a outra; após uma base sólida pronta, seria mais rápido escalar o projeto e implementar novos recursos.

A partir dessas considerações, optou-se por desenvolver o projeto em Python, com auxílio da IDE PyCharm.

4. Desenvolvimento

A primeira etapa para se iniciar o desenvolvimento foi escolher um pacote que resolvesse funções voltadas para a Teoria de Grafos. As principais bibliotecas disponíveis no Python são *NetworkX* [Aric et al. 2008], *Grinpy* [Amos 2018] e *Numpy* [Harris et al. 2020]. Para o desenvolvimento de Interface de Usuário, foi feita uma pesquisa para decidir qual opção se encaixava para o melhor desenvolvimento do projeto, assim como um estudo dessa biblioteca e o desenho das telas do programa que seriam implementadas. Por fim, foi necessário encontrar uma biblioteca para o esboço dos grafos que também permitisse a movimentação dos vértices pela tela.

Interface

Para a implementar a interface foi decidido por utilizar o *PyQt* [Riverbank Computing 2012], uma biblioteca feita em Python que é baseada na biblioteca *Qt* para C++. Sua escolha foi motivada por sua vasta comunidade, que inclui desenvolvedores de C++ e de Python. Portanto, caso algum suporte fosse necessário para o auxílio no desenvolvimento, seria fácil encontrar a resposta através da internet, por meios de fóruns ou da documentação oficial.

Antes de começar a criar os códigos da interface, em um primeiro momento foi estudado a biblioteca para entender de seus principais componentes e funcionamento, enquanto simultaneamente, o design das telas do programa foi revisto e melhorado, do ponto de vista a se parecer mais com um programa completo.

Equações e Invariantes

Para a criação das (in)equações, foi utilizado a biblioteca *SimpleEval* [Fairhead 2019], focada em interpretar expressões e traduzi-las para funções em Python. A partir desta, conseguimos chamar as funções de filtragem nas equações construídas de forma intuitiva, o que facilita o entendimento do usuário. Foram adicionadas funções lógicas ('AND' e 'OR') e funções usuais de matemática, que não tem suporte nativo pela biblioteca.

Os invariantes utilizados na construção das (in)equações e condições lógicas foram implementados pelas bibliotecas *NetworkX*, *Grinpy* e *Numpy*.

Desenho dos grafos

Foram necessárias muitas pesquisas para encontrar uma biblioteca para esboço dos grafos, pelo motivo de que as encontradas não permitiam a evolução do projeto por não suportarem qualquer tipo de edição nos vértices e arestas, ou não serem compatíveis com a linguagem de desenvolvimento do programa. Enfim foi encontrada a biblioteca *PyQtGraph* [PyQtGraph 2020], que tem compatibilidade não somente com a linguagem, mas também com a biblioteca de interface de usuário, permitindo ainda reimplementações para que novas funcionalidades sejam adicionadas.

Publicação do código

Este é um software de código aberto sob a licença *GNU General Public License v3.0*, desenvolvido e publicado no repositório [GraphFilter 2020] na plataforma GitHub, release 2.0. As distribuições da versão mais atual do programa, para Windows, Linux e Mac podem ser encontradas tanto no GitHub quanto no site oficial do software [Graph Filter webpage 2020] e juntamente a eles consta um guia rápido de como utilizar o software e instalar, além de informações sobre os desenvolvedores. Um guia mais completo com todas as informações do projeto pode ser encontrado na Wiki do repositório no GitHub, descrevendo todas as funcionalidades, detalhes sobre o desenvolvimento, lista de invariantes, usabilidade, auxílio na instalação e contribuição no código.

Vale ressaltar que a plataforma GitHub também oferece alternativas para que os usuários do programa deixem suas dúvidas e solicitações para implementação de novas funções, através da utilização do recurso de *Issues*, onde a pessoa cria uma tarefa e descreve o que deve ser implementado, ou ainda a aba de *Discussões*, um fórum onde os usuários podem debater sobre as utilizações do sistema e futuras implementações do programa.

5. Evolução

Com a nova implementação do projeto, muitos aspectos da versão anterior foram reavaliados, sempre pensando na melhor adaptação do usuário na utilização do programa.

Versão 1.0

Como dito anteriormente, o projeto em C# estava disponível apenas para o Sistema Operacional Microsoft Windows. Abrindo o projeto, pode ser visto uma única tela dividida em duas abas: *Filter Graphs*, onde o usuário colocaria as condições da filtragem, o método da filtragem e as equações, além de um dicionário descrevendo como se escrever uma equação; e *Visualization*, onde cada grafo de saída pode ser visto e estudado mais a fundo. Mais detalhes sobre a versão 1.0 são vistas no artigo [Ribeiro e Jones 2020].

Versão 2.0

Esta nova versão pode ser executada em qualquer Sistema Operacional. Abrindo o projeto, encontra-se uma tela inicial onde o usuário pode escolher criar ou abrir um projeto. Optando-se por abrir um projeto, uma janela de seleção de arquivo se abre permitindo que o usuário reveja os resultados de filtragem realizada anteriormente.

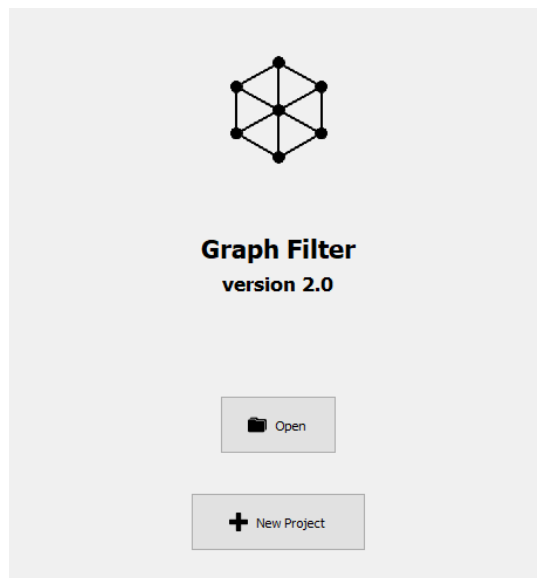


Figura 1 – Página principal Graph Filter

Em um novo projeto, o usuário entra em um *Wizard* que contém as seções necessárias para a filtragem dos grafos desejados. Algumas das páginas do *Wizard* contém um botão de ajuda com um símbolo de “?” que, passando o cursor do mouse por cima, dá ao usuário uma dica de como preencher a página em questão. Descreveremos a seguir cada etapa do processo de customização da filtragem pelo usuário.

Criação do projeto

A primeira página contém um campo para colocar o nome do projeto, assim como para escolher o caminho onde o projeto será salvo:

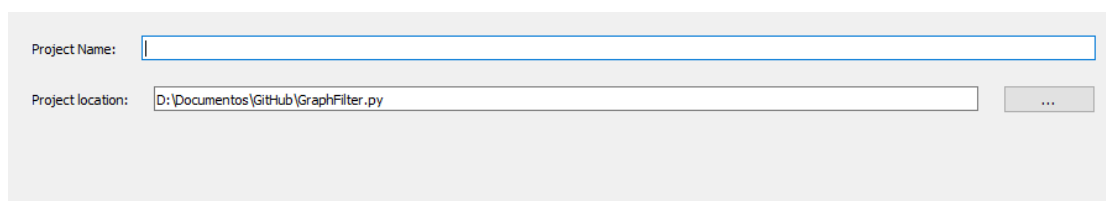


Figura 2 - Primeira página de Novo projeto

Inserção de equações e inequações

Nesta seção, o usuário pode inserir (in)equações contendo invariantes relacionados a grafos e operadores matemáticos, utilizando tanto a escrita através do teclado, quanto o auxílio de botões contidos em um bloco dividido por abas. Cada botão contém o nome de um invariante ou operação que, ao clicar, insere na caixa de texto o símbolo correspondente.

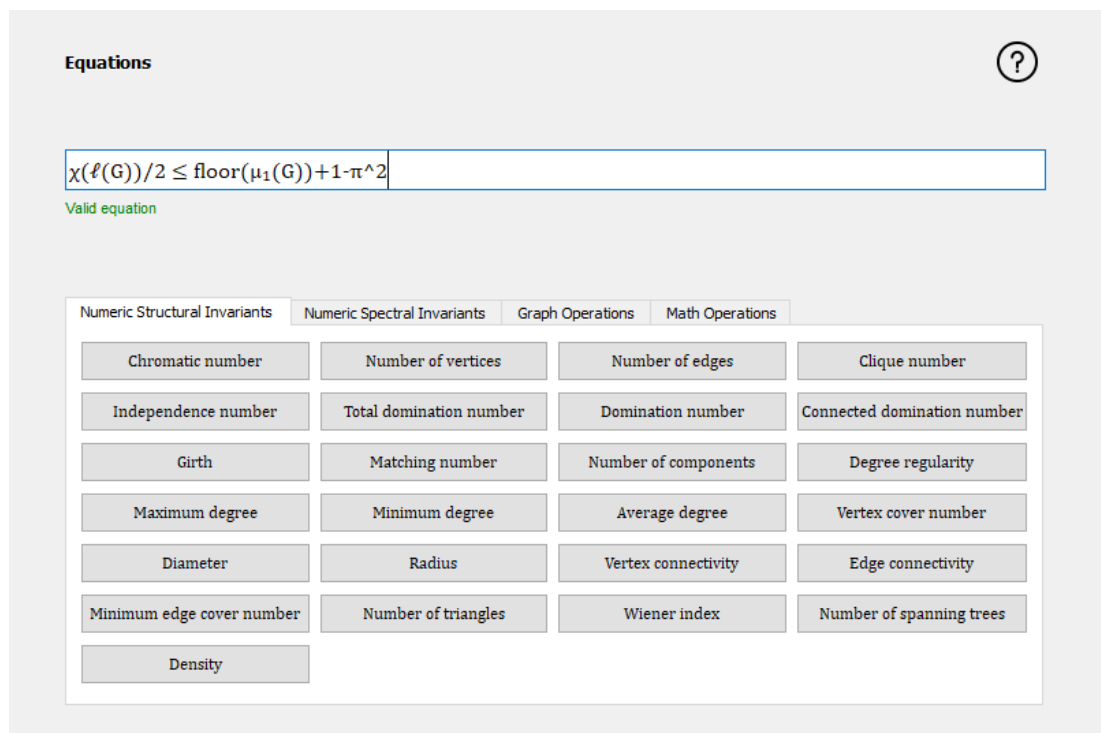


Figura 3 - Segunda página de Novo Projeto

A primeira e segunda abas são de invariantes numéricos, subdivididos em estruturais e espectrais, que podem ser aplicados em grafos ou compostos com outras operações. Importante ressaltar que nesta versão contamos com 55 invariantes, em contraste com a versão 1.0, que contava com 22.

Estrutural (<i>Structural</i>)		Espectral (<i>Spectral</i>)
Chromatic number	Maximum degree	Largest M-eigenvalue
Number of vertices	Minimum degree	Estrada index
Number of edges	Average degree	Number of main M-eigenvalues
Clique number	Vertex cover number	Second Largest M-eigenvalue
Independence number	Diameter	Nullity
Total domination number	Radius	Rank of matrix M
Domination number	Vertex connectivity	Algebraic connectivity
Connected domination number	Edge connectivity	M-Energy
Number of Spanning trees	Girth	Determinant of matrix M
Matching number	Number of triangles	
Number of components	Wiener Index	
Degree regularity	Density	
Minimum edge cover number		

Onde M pode ser as matrizes:
Adjacência, Laplaciana, Laplaciana sem sinal, Laplaciana Normalizada ou Distância.

Tabela 1 - Invariantes numéricos

A terceira aba contém as operações em grafos, que permitem que o usuário calcule os invariantes não apenas no próprio grafo presente na sua lista de entrada, mas também pelas operações *Complement*, *Line Graph* ou *Clique Graph (maximal)*.

Finalmente, a quarta aba contém as funções usadas em matemática, tais como: piso, teto, módulo, funções trigonométricas, congruência modular, dentre outras.

Por exemplo, se o usuário deseja que a filtragem satisfaça a condição

$$\omega(\ell(G)) \leq (\omega(G))^2 + \lambda_1(\overline{G}),$$

onde ω é o número clique, λ_1 é o raio espectral do grafo e ℓ a operação grafo linha, então nesta versão 2.0 a caixa de texto deve ser preenchida por

$$\omega(\ell(G)) \leq \omega(G)^2 + \lambda_1(Comp(G)),$$

reforçando que a inserção dos símbolos pode ser feita com auxílio dos botões ou digitando o nome do invariante, onde este será convertido para a simbologia usual, garantindo uma escrita intuitiva ao usuário. Em comparativo com a versão 1.0, a mesma inequação seria descrita por

$$omega_lg \leq (omega_G)^2 + lambda1_cG.$$

Para observar a evolução da eficiência na nova versão, foi realizado um teste comparativo simples, que consistiu em filtrar todos os grafos de 8 vértices que satisfazem a inequação acima. Constatamos que a versão 2.0 realizou a filtragem aproximadamente 2,15 vezes mais rápido que na versão anterior, para esta instância.

Condições lógicas

Na terceira página do *Wizard*, o usuário pode escolher as condições lógicas da filtragem.

Figura 4 - Terceira página de Novo Projeto

Para tal, utilizamos os invariantes lógicos, onde o usuário pode marcar 'true', se deseja que os grafos satisfaçam a quela condição, 'false', se deseja que os grafos não satisfaçam aquela condição, ou ainda deixar em branco para que aquela condição seja irrelevante na filtragem. Isto representa uma evolução da versão anterior, que só permitia marcar condições para serem satisfeitas ou desconsideradas. Além disso, vale ressaltar que nessa versão contamos com 35 invariantes lógicos, enquanto a anterior possuía 15.

Estrutural (<i>Structural</i>)			Espectral* (<i>Spectral</i>)
Planar	Chordal	Self-complementary	Some M-eigenvalue integer
Connected	Triangle-free	Cubic	M-integral
Biconnected	Regular	Has bridge	Largest M-eigenvalue integer
Bipartite	Claw-free	Regular transmission	Matrix M is invertible
Eulerian	Tree	Bull-free	

*Onde M pode ser as matrizes:
Adjacência, Laplaciana, Laplaciana sem sinal, Laplaciana Normalizada ou Distância.

Tabela 2 - Invariantes lógicos

Método de Filtragem

Na quarta página, o usuário pode escolher o método de filtragem por *Filter Graphs* ou *Find Counterexample*. A primeira opção vai analisar cada grafo e descartar da lista aqueles que não satisfazem as condições impostas pelo usuário, enquanto que a segunda opção vai parar a busca no primeiro grafo que não satisfazer as condições, caso exista. Esses métodos foram mantidos como os da versão anterior.

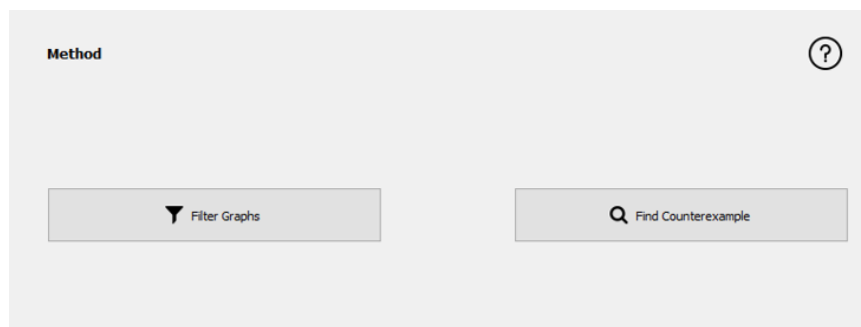


Figura 5 - Quarta página de Novo Projeto

Inserção dos grafos de entrada

Na quinta página do *Wizard*, o usuário pode inserir os arquivos de grafo no formato *graph6* [Mckay 2007] que serão utilizados na filtragem. O usuário não precisa ter conhecimento sobre este formato, pois diversas páginas disponibilizam classes de grafos em *.g6*, como por exemplo em [Brinkmann et al. 2013]. Há inclusive um botão “*get .g6 graph*” onde o usuário será levado diretamente a um site onde ele pode escolher por uma classe de grafos disponibilizadas, e efetuar o *download* no formato *.g6*. Diferentemente da versão anterior, agora é permitido utilizar vários arquivos *.g6* ou *.txt* para serem inseridos na filtragem, clicando no botão de “+”. Vale ainda reforçar que nesta nova versão é permitido também arquivos compactados por *gzip*, ou seja, os formatos *.g6.gz* ou *.txt.gz* também são aceitos.



Figura 6 - Quinta página de Novo Projeto

Revisão das configurações de filtragem

Antes de finalmente fazer a filtragem, é exibido uma tela de revisão para que o usuário confirme as opções que marcou durante o processo do novo projeto.

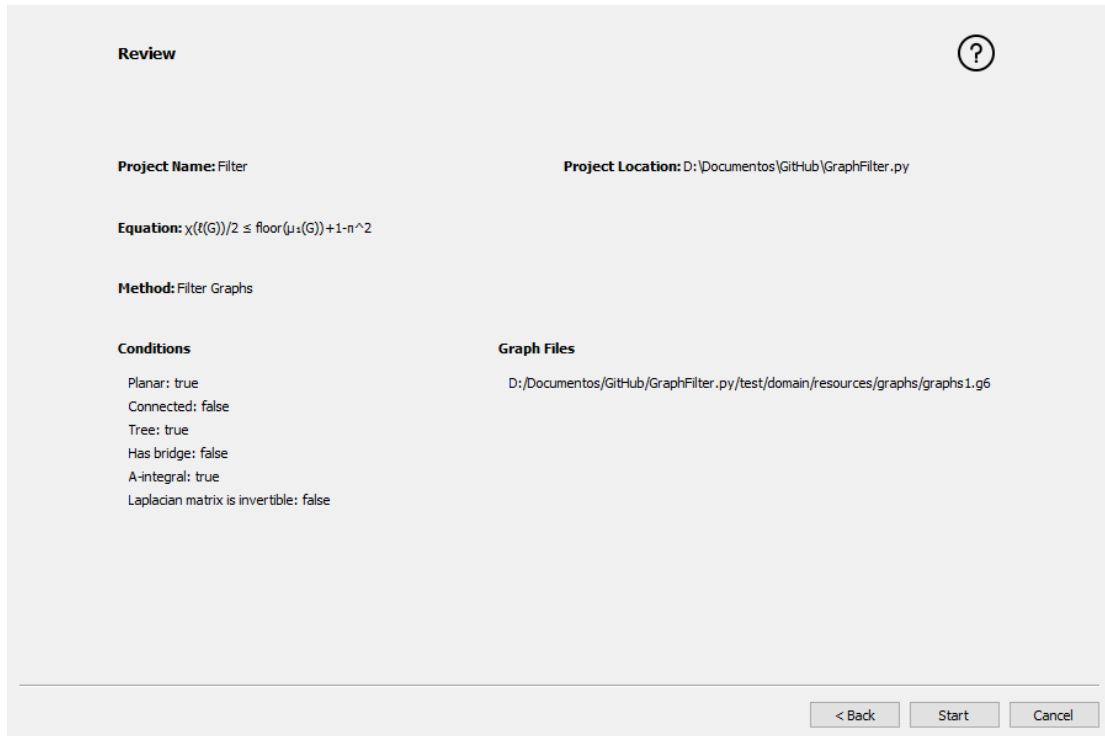


Figura 7 - Sexta página de Novo Projeto

Ao clicar no botão de *Start*, uma tela de carregamento aparece e mostra ao usuário a porcentagem dos grafos que foram processados até o momento. Assim que a tela de carregamento se encerra, a tela principal do projeto se abre.

Tela principal do projeto

Nesta janela o usuário poderá visualizar e interagir com todos os grafos retornados pela filtragem, onde terá ainda acesso à diversos invariantes que podem ser calculados para o grafo esboçado na tela, possibilitando uma análise aprimorada para seu estudo dos grafos retornados.

Fixo no topo da janela principal, há um *toolbar* que contém uma *combo-box* onde se encontra todos os grafos resultantes da filtragem. As setas para a direita e esquerda facilitam a navegação entre um grafo e outro. O conteúdo da janela principal é dividido em abas, sendo elas: *Visualize*, *Invariants*, *Dictionary* e *Info*. Todas essas abas podem ser movidas de um lugar para o outro, e podem ter seu tamanho alterado, com algumas limitações. Também podem ser separadas do corpo da janela, ou ainda fechadas caso não estejam sendo utilizadas. Para reabri-las, basta clicar no nome da aba correspondente através da barra de menu, na opção de *View*.

A aba *Visualize* contém uma visualização interativa do grafo selecionado na *combo-box* da *toolbar*. É possível mover os vértices para melhor visualização, além de possibilitar aproximar e afastar a tela. Clicando com o botão direito do mouse em cima do grafo, é possível modificar alguns controles de interação, e exportar o grafo esboçado para diversos formatos.

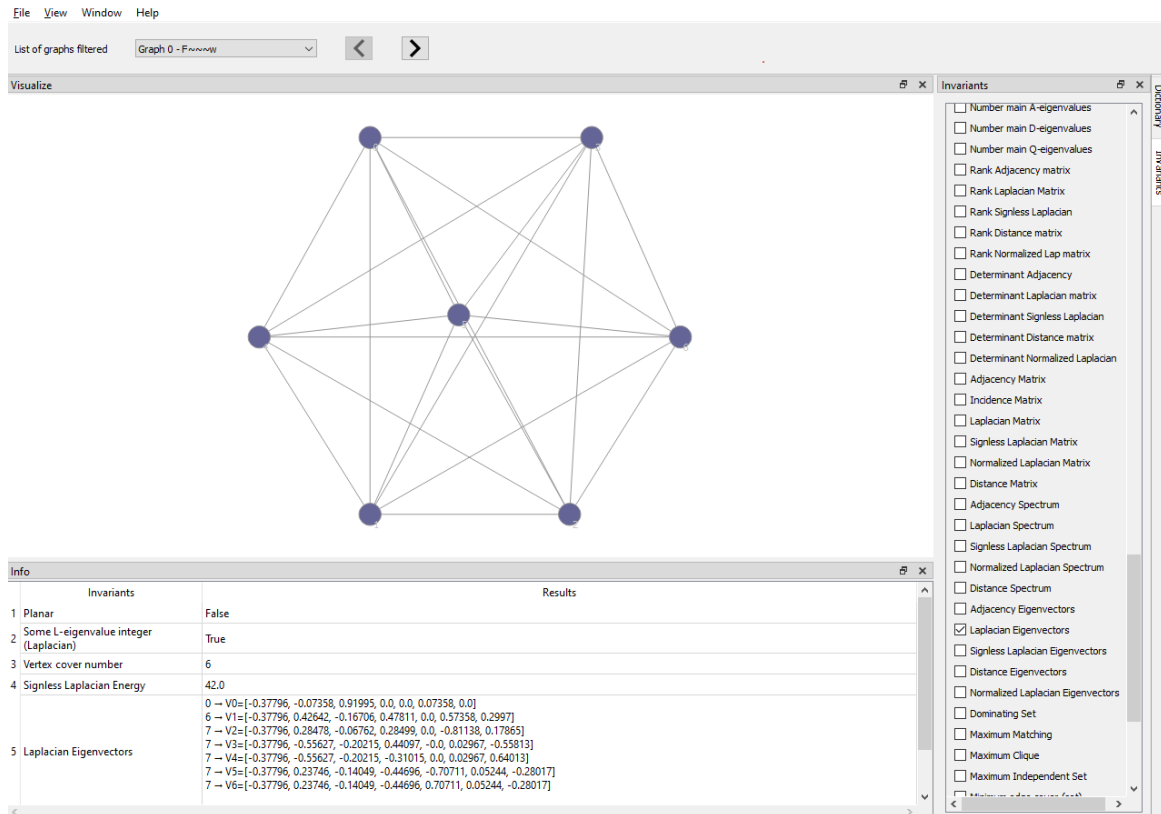


Figura 8 - Janela principal do programa com aba Invariants

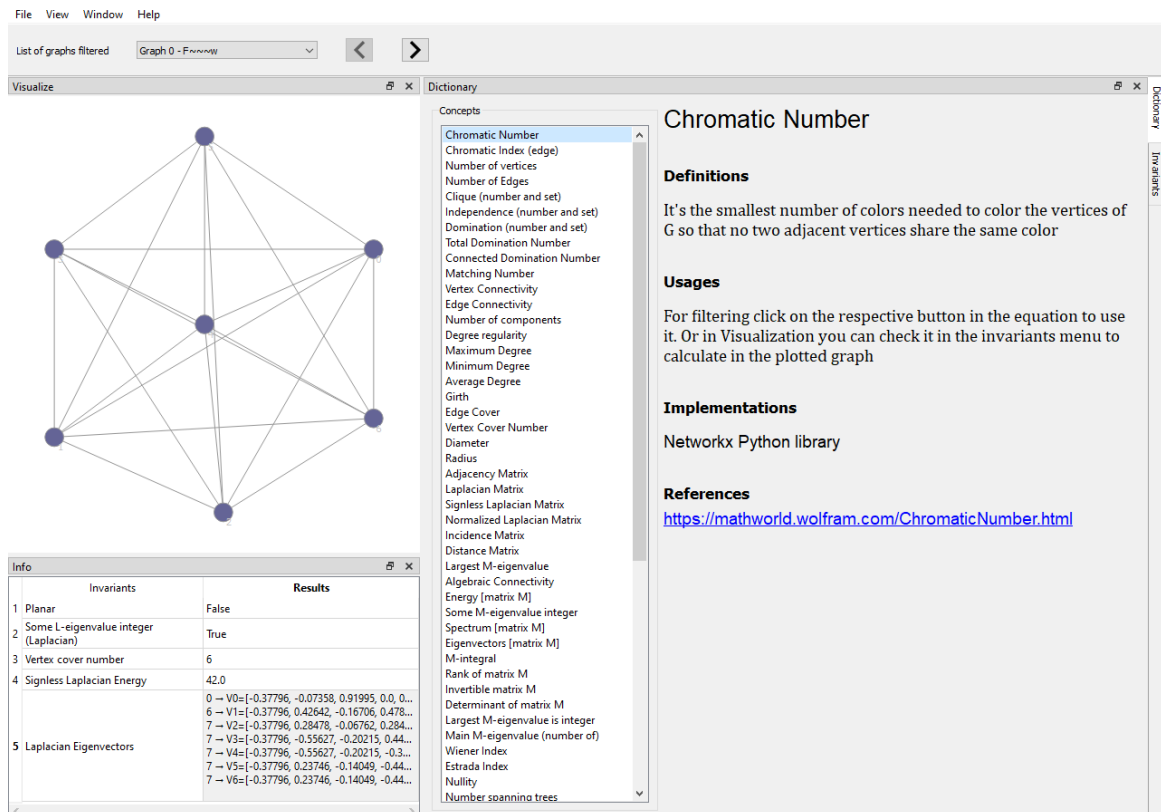


Figura 9 - Janela principal do programa com aba Dictionary

Na aba *Invariants* existe uma lista de invariantes que podem ser calculados baseando-se no grafo selecionado. Essa lista contém todos os invariantes vistos anteriormente, além de alguns extras, que não são invariantes numéricos ou lógicos, pois fornecem como retorno matrizes, conjuntos ou vetores:

Estrutural (<i>Structural</i>)	Estruturais* (<i>Spectral</i>)
M-Matrix	Dominating Set
M-Spectrum	Maximum Matching Set
M-Eigenvectors	Maximum Clique
	Maximum Independent Set
	Minimum edge cover (set)
<i>*Onde M pode ser as matrizes: Adjacência, Laplaciana, Laplaciana sem sinal, Laplaciana Normalizada ou Distância.</i>	

Tabela 3 - Invariantes Extras

O resultado dos cálculos para o grafo selecionado de acordo com a lista de invariantes fica disponível na aba *Info* em formato de tabela.

A aba *Dictionary* possui uma lista dos invariantes do programa, e selecionando cada uma é possível ver uma descrição breve, a referência contendo um link de acesso, e a respectiva biblioteca que foi utilizada em sua implementação.

O software também possui opções para exportar todos os grafos resultantes da filtragem, nos seguintes formatos: PDF, PNG, TikZ e G6, onde os dois primeiros foram implementados por *Matplotlib* [Hunter 2007], enquanto TikZ por *network2tikz* [Hackl 2018].

6. Considerações Finais

Como visto, este trabalho apresenta evoluções significativas em relação à versão anterior, com melhoramento da interface de usuário, aumento na quantidade de invariantes e de funcionalidades, implementação de dicionário, escrita de equações de forma mais intuitiva, e aperfeiçoamento da arquitetura possibilitando incrementos na aplicação.

O projeto continua em desenvolvimento, e nele ainda será implementado: a paralelização da filtragem; possíveis novas condições de filtragem; e permitir o usuário a editar os grafos resultantes da filtragem.

Agradecimento

Agradecemos ao Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais, Campus Juiz de Fora, pelo apoio financeiro para desenvolvimento deste trabalho.

Referências

Amos, D. (2018). Grinpy: A NetworkX extension for calculating graph invariants. Web page: <https://github.com/somacdivad/grinpy>. GitHub repository. Acessado: 18-05-2021

Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, (2008). Exploring network structure, dynamics, and function using NetworkX”, in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15.

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

Fairhead, D. (2019). Simpleeval (Simple Eval). Web page: <https://github.com/danthedeckie/simpleeval>. Acessado: 06-06-2021

Graph Filter webpage (2020). Graph Filter. Webpage: <http://sistemas.jf.ifsudestemg.edu.br/graphfilter/home>. Acessado: 06-06-2021.

GraphFilter (2020). Graph Filter: Software to help Graph researchers providing filtering and visualization to a given list of graphs. Web page: <https://github.com/GraphFilter/GraphFilter>. Acessado: 06-06-2021.

Hackl, J. (2018). network2tikz. Web page: <https://github.com/hackl/network2tikz>. GitHub repository. Acessado: 31-05-2021

Harris, C. R., Millman, K. J., van der Walt, S. J. et al. (2020). Array programming with NumPy. *Nature*. September, v. 585, n. 7825, p 357-362.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, IEEE COMPUTER SOC. v. 9, n 3, p. 90-95.

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

McKay, B. D. (2007). Web page. <http://users.cecs.anu.edu.au/~bdm/data/graphs.html>. Acessado: 18-05-2021

PyQtGraph (2020). PyQtGraph: A pure-Python graphics library for PyQt5. Web page: <https://github.com/pyqtgraph/pyqtgraph>. GitHub repository. Acessado: 18-05-2021

Ribeiro, D. P. de O. e Jones, A. A. (2020). Graph Filter: um software intuitivo para filtragem de grafos com propriedades customizadas pelo usuário. In *Anais do X ERMAC-RS*, Rio Grande do Sul, ediPUCRS.

Riverbank Computing (2012). PyQt5. Web page: <https://www.riverbankcomputing.com/software/pyqt>. Acessado: 18-05-2021

Zapponi, C. (2020). GitHut 2.0 - A small place to discover languages in GitHub. Web page madnight.github.io/githut. Acessado: 18-05-2021