Teoria Espectral de Grafos Aplicada ao Problema de Isomorfismo de Grafos

Philippe Leal Freire dos Santos
Maria Cristina Rangel
Maria Claudia Silva Boeres
Universidade Federal do Espírito Santo - UFES
Departamento de Informática
Av. Fernando Ferrari, 514 - Goiabeiras - 29075-910 - Vitória - ES - Brasil
philippeleal@gmail.com, {crangel, boeres}@inf.ufes.br

RESUMO

Teoria Espectral de Grafos consiste em uma área de estudo que relaciona propriedades de grafos com alguns conceitos específicos de álgebra linear. Por exemplo, conceitos como o espectro de um grafo e as centralidades de seus vértices tem sido usados em Teoria dos Grafos para caracterização e extração de informações estruturais relevantes. Neste trabalho, apresentamos alguns resultados teóricos que consideram informações do espectro e das centralidades dos vértices dos grafos. Um algoritmo para detecção de isomorfismo de grafos baseado nestes resultados é proposto.

PALAVRAS CHAVES. Problema de Isomorfismo de Grafos. Teoria Espectral de Grafos. Vetor de centralidades. Teoria dos Grafos.

ABSTRACT

Spectral graph theory is a knowledge area that studies graph properties by their relationship with some specific linear algebra concepts. For instance, concepts as graph spectrum and eigenvector centrality have been used in graph theory to characterize graph properties and extract relevant graph structural information. In this work some theoretic results and an algorithm using them to detect the isomorphism between two graphs considering their spectrum information and eigenvector centrality is proposed.

KEYWORDS. Graph isomorphism problem. Spectral Graph Theory. Eigenvector centrality. Graph Theory.

1 Introdução

O problema de isomorfismo de grafos (PIG) pode ser aplicado a diversos problemas práticos, como por exemplo, no reconhecimento de impressões digitais aplicados a sistemas de segurança (Ferreira Jr et al. (2001)) e na área de química, quando se quer determinar se uma molécula possui ou não estrutura similar a outra (Fortin (1996) e Oliveira e Greve (2005)), entre outros. O PIG consiste em encontrar uma correspondência um a um dos vértices de dois grafos dados, obedecendo as adjacências existentes entre os vértices (Berge (1991)).

Atualmente a sua complexidade ainda permanece como uma incógnita (Fortin (1996)). Este é um dos poucos problemas que pertence à classe dos problemas NP, do qual não se sabe se ele está na classe P ou NP-completo. Entretando, é conhecido que não se trata de um problema co-NP.

Devido a incerteza quanto à sua localização nas classes de complexidade, alguns algoritmos exatos para solucioná-lo na sua forma geral foram desenvolvidos, como o algoritmo de Ullmann (Ullmann (1976)), o algoritmo *Nauty* (McKay (1981)) e o algoritmo proposto por Lee (2007), que utiliza técnicas que extraem informações dos grafos de entrada, entre outros. Existem também exemplos de algoritmos de tempo polinomial, dedicados a classes específicas de grafos (Sorlin & Solnon (2004), Uehara et al. (2005) e Zager (2005)). Além disso, podemos citar Xiutang & Kai (2008) como exemplo de algoritmos heurísticos para resolver o PIG. Dharwadker & Tevet (2009) apresentaram um algoritmo que, segundo eles, possui tempo polinomial quando executado para qualquer classes de grafos.

A Teoria Espectral de Grafos (TEG) analisa propriedades de grafos através de matrizes e seus espectros. Este estudo pode ser realizado sobre várias matrizes de representação de um grafo, como por exemplo, a matriz de adjacência, a laplaciana e a laplaciana sem sinal. Conceitos como o espectro de um grafo e as centralidades de seus vértices tem sido usados em Teoria dos Grafos para caracterização e extração de informações estruturais relevantes.

Neste trabalho buscamos identificar conceitos na TEG que possam auxiliar na construção de algoritmos para detecção de isomorfismo de grafos. Apresentamos alguns resultados teóricos que consideram informações do espectro e das centralidades dos vértices dos grafos. Um algoritmo para detecção de isomorfismo de grafos baseado nestes resultados é proposto. Ele é aplicado a um conjunto de 3.000 pares de grafos conexos isomorfos gerados aleatoriamente a partir da biblioteca VFLib, que foi desenvolvida por Santo et al. (2003) com a finalidade de servir como *benchmark* para algoritmos que se propõem a solucionar o PIG. O seu tempo de execução foi comparado aos tempos de dois outros algoritmos, propostos por Lee (2007) e Dharwadker & Tevet (2009).

A seção seguinte trata do Problema de Isomorfismo de Grafos apresentando sua definição e mostrando exemplos. Na Seção 3 alguns conceitos da Teoria Espectral de Grafos são definidos e na Seção 4, dois resultados teóricos são apresentados, a fim de auxiliar o entendimento do algoritmo proposto na Seção 5. A Seção 6 é dedicada à análise dos testes computacionais. Por fim, na Seção 7, são apresentadas a conclusão e as perspectivas de futuros trabalhos.

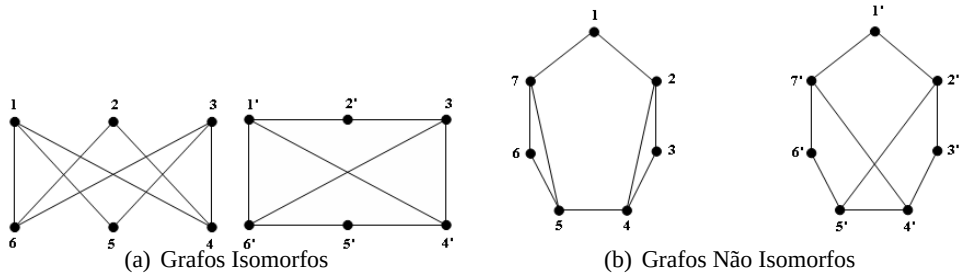
2 Problema de Isomorfismo de Grafos

O Problema de Isomorfismo de Grafos (PIG) tem sido amplamente estudado devido a sua grande aplicabilidade como modelo matemático para problemas reais em várias áreas do conhecimento. A área de Química é um exemplo disto, onde constantemente é necessário determinar se uma molécula posssui ou não estrutura similar a uma outra, para que se possa atribuir-lhe um nome exclusivo. Esta verificação pode ser realizada comparando a molécula a uma base existente de dados moleculares. Uma possível estratégia consiste em representar as moléculas por grafos, sendo os vértices correspondentes aos átomos e as arestas às suas ligações (Fortin (1996) e Oliveira & Greve (2005)). Desta forma, o processo de comparação entre as moléculas pode ser modelado

pelo PIG, uma vez que este problema consiste em verificar se dois grafos dados são estruturalmente idênticos, ou seja, se existe um mapeamento um a um entre os vértices do primeiro grafo com os do segundo que preserve as relações de adjacência.

Formalmente, diz-se que dois grafos $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$ são ditos isomorfos se existe uma função bijetora $f : V_1 \rightarrow V_2$ tal que, $\forall a, b \in V_1$, $(a, b) \in E_1 \Leftrightarrow (f(a), f(b)) \in E_2$. O PIG consiste em determinar se dois grafos são isomorfos. Um exemplo de dois grafos isomorfos pode ser visto na Figura 1(a), onde é possível encontrar uma função $f : V_1 \rightarrow V_2$, $f = \{(1, 1'), (2, 5'), (3, 3'), (4, 4'), (5, 2'), (6, 6')\}$, que mantém as adjacências existentes entre os vértices dos grafos.

Em decorrência da definição do PIG, para que dois grafos sejam isomorfos, eles têm que possuir o mesmo número de vértices e de arestas, e mesma sequência de graus dos vértices (Dalcumune (2008)). Embora necessárias, estas condições não são suficientes para o isomorfismo entre dois grafos. Como exemplo, temos os grafos da Figura 1(b) que atendem estas condições, porém não são isomorfos.



(a) Grafos Isomorfos (b) Grafos Não Isomorfos

Figura 1: Exemplos de grafos isomorfos e não isomorfos

3 Teoria Espectral de Grafos

A Teoria Espectral de Grafos (TEG) é uma parte da matemática discreta que estuda as propriedades de um grafo a partir das informações fornecidas pelo espectro da matriz associada a este grafo, por exemplo, a matriz de adjacência (que utilizaremos neste trabalho) e a matriz laplaciana, Hogben (2009). A TEG tem despertado interesse de estudiosos nas últimas três décadas devido a sua aplicação em várias áreas, como na química, na engenharia e na ciência da computação (Abreu (2005)). Nesta seção serão apresentadas algumas definições referentes à TEG, extraídas de Abreu et al. (2007), que auxiliarão no entendimento de assuntos tratados neste trabalho.

Definição 1 *Seja $G = (V, E)$ um grafo simples, não direcionado com n vértices e m arestas. A matriz $n \times n$ cujas entradas são iguais a 1, se u e v são adjacentes, e 0 caso contrário, com u e $v \in V$, é denominada matriz de adjacência de G.*

Definição 2 *O polinômio característico da matriz de adjacência $A(G)$ de um grafo G é chamado polinômio característico de G e denotado por $p_G(\lambda)$. Assim, $p_G(\lambda) = det(A(G) - \lambda I)$, onde λ é uma raiz deste polinômio e dito ser um autovalor de G. Tendo o grafo n vértices, consequentemente ele possui n autovalores, sendo o maior deles denominado índice do grafo.*

Definição 3 *O espectro de G, indicado por $spect(G)$, é definido como uma matriz $2 \times d$, tendo na primeira linha os d autovalores distintos de G dispostos em ordem decrescente e na segunda linha as suas respectivas multiplicidades algébricas.*

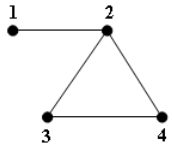


Figura 2: Grafo G_1

Como exemplo, observe o grafo G_1 da Figura 2. O seu polinômio característico é $p_{G_1}(\lambda) = \lambda^4 - 4\lambda^2 - 2\lambda + 1$, tendo como espectro:

$$spect(G_1) = \begin{bmatrix} 2,1701 & 0,3111 & -1 & -1,4812 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Definição 4 *Dois grafos G_1 e G_2 são chamados co-espectrais quando seus autovalores são iguais considerando suas multiplicidades, ou seja, quando $spect(G_1) = spect(G_2)$.*

Como resultado desta definição temos que, se dois grafos são isomorfos, então eles têm o mesmo espectro. Entretanto, a recíproca desta afirmação não é sempre verdadeira. Para exemplificar, considere os grafos G_1 e G_2 da Figura 3. Estes grafos possuem o mesmo polinômio característico, $p_{G_1}(\lambda) = p_{G_2}(\lambda) = \lambda^6 - 7\lambda^4 - 4\lambda^3 - 7\lambda^2 + 4\lambda - 1$, portanto são co-espectrais, porém não isomorfos.

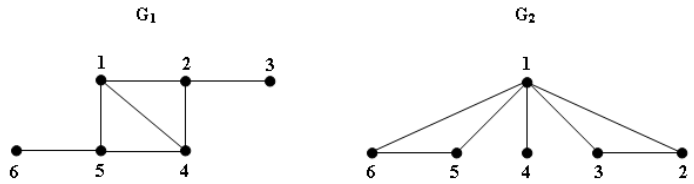


Figura 3: Grafos co-espectrais, porém não isomorfos (Extraídos de Abreu (2005))

Definição 5 *Dada uma matriz A, um vetor não nulo v tal que $Av = \lambda v$ é dito autovetor associado ao autovalor λ.*

Definição 6 *A centralidade de autovetor x_i é definida como a i-ésima componente do autovetor não-negativo x associado ao índice do grafo, $i = 1, \ldots, n$.*

Exemplificando, seja o grafo G_1 da Figura 2, que tem índice igual a $2,1701$. Um autovetor associado a este índice é: $[0,2818,\ 0,6116,\ 0,5227,\ 0,5227]$. Portanto, cada vértice i de G_1 tem uma centralidade de autovetor x_i associada a ele, para uma dada ordenação dos seus vértices.

4 Alguns Resultados Teóricos

Uma das motivações deste trabalho consiste no uso de elementos da Teoria Espectral de Grafos, especialmente autovalores e autovetores, em algoritmos para detecção do isomorfismo de grafos. Mais precisamente, estudamos a centralidade dos vértices de um grafo como um possível filtro adicional na detecção de grafos isomorfos, uma vez que resultados existentes na literatura afirmam que dois vértices em um mesmo grafo, que possuam mesmo grau, podem ter centralidades distintas

(Grassi et al. (2007)). Assim, apresentamos dois resultados teóricos que foram utilizados no algoritmo proposto na Seção 5. O primeiro deles foi provado como o Teorema 1 e diz respeito a grafos com vetores de centralidades iguais, porém com componentes distintas entre si. O teorema afirma que dois grafos com essas características são isomorfos. O segundo resultado consiste em uma conjectura, baseada nos experimentos realizados com grafos isomorfos e não isomorfos. Em todos os casos de isomorfismo, os vetores de centralidades de ambos eram proporcionais. Nos casos de não isomorfismo, observamos que essa característica não se verificava. Esse resultado é enunciado em Conjectura 1.

Teorema 1 *Sejam G_1 e G_2 dois grafos simples, conexos e com mesmo índice. Se seus vetores de centralidades forem proporcionais e as componentes de cada vetor forem distintas entre si então os grafos são isomorfos.*

Prova:

Sejam $\vec{x}^1$ e $\vec{x}^2$ autovetores positivos associados, respectivamente, aos índices dos grafos G_1 e G_2. Suponha que $|V_1| = |V_2| = n$, $|E_1| = |E_2| = m$ e que $\vec{x}^i = (x_1^i, ..., x_n^i)$, tal que $x_j^i \neq x_k^i$, $j, k = 1, \ldots, n$, $j \neq k$ e $i = 1, 2$. Suponha ainda que $\vec{x}^1 = \vec{x}^2$ para alguma ordenação dos componentes de $\vec{x}^i, i = 1, 2$. Pela definição de centralidade de autovetor, $\vec{x}^i$ define as centralidades de autovetor associadas aos vértices dos grafos G_i para alguma ordenação de V_i, $i = 1, 2$. Desta forma supomos que G_1 e G_2 possuem as mesmas centralidades de autovetor para alguma ordenação de seus conjuntos de vértices. Seja

$$f : V_1 \to V_2 \text{ uma função biunívoca, tal que } f(v) = w \text{ se, e somente se, } c(v) = c(w) \tag{1}$$

onde $v \in V_1$, $w \in V_2$ e $c(\cdot)$ representa a centralidade do vértice. Assim,

$$c(v) = \vec{x}_j^1 = \vec{x}_k^2 = c(w), \text{ para algum } j, k \in \{1, \ldots, n\} \tag{2}$$

Em Bonacich (2007) temos que

$$\lambda x_i = \sum_{j=1}^{n} a_{ij} x_j, i = 1, \ldots, n, \text{ onde } a_{ij} \in A(G) \text{ e } \lambda \text{ é o índice de } G. \tag{3}$$

Desta forma, podemos afirmar que a centralidade de um vértice pode ser obtida a partir do somatório das centralidades de seus vizinhos dividido pelo índice do grafo. Assim, reescrevendo (3) na forma de matriz de G_1, temos

$$\underbrace{\begin{bmatrix} a_{11}^1 & a_{12}^1 & \cdots & a_{1n}^1 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1}^1 & a_{n2}^1 & \cdots & a_{nn}^1 \end{bmatrix}}_{A(G_1)} \begin{bmatrix} x_1^1 \\ x_2^1 \\ \vdots \\ x_n^1 \end{bmatrix} = \lambda \begin{bmatrix} x_1^1 \\ x_2^1 \\ \vdots \\ x_n^1 \end{bmatrix} \tag{4}$$

onde $A(G_1)$ é a matriz de adjacência de G_1 e, portanto, $a_{ii} = 0, i = 1, \ldots, n$.

Considere agora uma reordenação de $\vec{x}^2$ dada pela função f de acordo com (1) originando $\vec{x}^{2'}$. Sendo G_1 e G_2 de mesmo índice, $\lambda_1^1 = \lambda_1^2 = \lambda$. Então, $\lambda\vec{x}^1 = \lambda\vec{x}^2$. Daí e de (3) temos

$$\underbrace{\begin{bmatrix} a_{11}^2 & a_{12}^2 & \cdots & a_{1n}^2 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1}^2 & a_{n2}^2 & \cdots & a_{nn}^2 \end{bmatrix}}_{A'(G_2)} \begin{bmatrix} x_1^{2'} \\ x_2^{2'} \\ \vdots \\ x_n^{2'} \end{bmatrix} = \lambda \begin{bmatrix} x_1^{2'} \\ x_2^{2'} \\ \vdots \\ x_n^{2'} \end{bmatrix} \tag{5}$$

Assim, de (4) e (5) temos que

$$\begin{bmatrix} a_{11}^1 & a_{12}^1 & \cdots & a_{1n}^1 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1}^1 & a_{n2}^1 & \cdots & a_{nn}^1 \end{bmatrix} = \begin{bmatrix} a_{11}^2 & a_{12}^2 & \cdots & a_{1n}^2 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{n1}^2 & a_{n2}^2 & \cdots & a_{nn}^2 \end{bmatrix} \tag{6}$$

Ou seja, $A(G_1) = A^{'}(G_2)$, onde $A^{'}(G_2)$ decorre de uma permutação de linhas e colunas de $A(G_2)$ dada pela função f. Desta forma, todas as arestas de G_1 se preservam em G_2. Assim, $(v, u) \in E_1$ se, e somente se, $(f(v), f(u)) \in E_2$. Então, por definição, a função f define um mapeamento entre V_1 e V_2 que é um isomorfismo. Logo, G_1 e G_2 são isomorfos. ■

Conjectura 1 *Se dois grafos são isomorfos então suas centralidades de autovetor são proporcionais.*

5 Algoritmo para Detecção de Isomorfismo de Grafos

O PIG pertence à classe de problemas NP, entretanto é desconhecido se está em P ou em NP-completo (Jenner et al. (2003)). A suposição comumente aceita é que ele esteja estritamente entre estas duas classes (Arvind & Torán (2005)). Apesar disto, foram desenvolvidos algoritmos de tempo polinomial dedicados a muitas classes de grafos. Em Sorlin & Solnon (2004), Uehara et al. (2005) e Zager (2005) são citadas algumas destas classes, que incluem grafos planares, grafos convexos, grafos limitados por grau, árvores, entre outros.

Também foram implementados algoritmos exatos para solucionar o problema na sua forma geral, como o algoritmo de Ullmann (1976), o algoritmo *Nauty* (McKay (1981)), o *VF* (Cordella et al. (1999)), e sua segunda versão denominada *VF2* (Cordella et al. (2001)), e o proposto por Lee (2007). Não obstante a incerteza quanto à complexidade do PIG, Dharwadker & Tevet (2009) apresentaram um algoritmo que, segundo eles, executa em tempo polinomial para todas as classes de grafos e é necessário e suficiente para solucionar o problema. Porém, em nossos testes, encontramos um par de grafos que torna falsa esta afirmação (ver Seção 6).

Embora outros algoritmos que solucionam o problema em seu caso geral já tenham sido propostos, neste trabalho apresentamos um novo algoritmo baseado em propriedades da Teoria Espectral de Grafos, com utilização de dois filtros: um exato e outro heurístico. Entende-se como filtro uma condição que torna mais simples a busca pela resposta do problema. O filtro heurístico é baseado na Conjectura 1. Já o filtro exato, baseado no Teorema 1, evita a descida na árvore de busca de soluções.

O algoritmo proposto pode ser dividido em três fases, as quais são descritas a seguir (claramente, a primeira fase só é executada quando os grafos possuem o mesmo número de vértices, de arestas e a mesma sequência de graus).

5.1 Fase 1: Cálculo dos índices dos grafos e do autovetor associado

Nesta fase realizamos o cálculo dos índices (λ_1 e λ_2) e seus autovetores associados ($\vec{x}^1$ e $\vec{x}^2$), referentes aos dois grafos de entrada. Lembrando que cada componente x_j^i, $i = 1, 2$, do autovetor associado é a centralidade do vértice v_j do grafo, para alguma ordenação dos rótulos de seus vértices. A intenção na utilização destes autovetores é reunir os vértices dos grafos em *blocos* de centralidades proporcionais. Com isso, a tentativa de associação fica restrita a vértices de um mesmo bloco, objetivando assim reduzir possíveis soluções inviáveis para o problema.

Para esse fim, ordenamos as centralidades de maneira crescente e as comparamos proporcionalmente. Caso a centralidade x_j^1 do grafo G_1 for proporcionalmente igual à centralidade x_j^2 do grafo

G_2, onde $j = 1, \ldots, n$, o algoritmo segue para a Fase 2, uma vez que há a possibilidade dos grafos serem isomorfos. Caso contrário, consideramos estes grafos não isomorfos. Essa decisão é baseada na Conjectura 1, caracterizando o filtro heurístico para o algoritmo.

5.2 Fase 2: Verificação da distinção das centralidades de um mesmo autovetor

Ao chegar nesta fase, os autovetores de ambos os grafos são iguais (a menos de uma constante) e ordenados de maneira crescente. O objetivo aqui é verificar se as centralidades de cada autovetor são distintas entre si. Se esse fato acontece, de acordo com o Teorema 1, os grafos são considerados isomorfos. Caso isto não ocorra, o algoritmo avança para a Fase 3, onde fará a descida na árvore de busca de soluções do PIG. Esta fase define o filtro exato do algoritmo.

É fácil perceber que se os autovetores possuem centralidades proporcionais, com componentes distintas entre si, cada vértice de ambos os grafos pertencerá a um único bloco de centralidades. Assim, o número de blocos será igual ao número de vértices dos grafos. Deste modo, cada vértice de um grafo só poderá ser associado a um único vértice do outro grafo, respeitando a igualdade dos graus.

5.3 Fase 3: Descida na árvore de busca

O algoritmo atinge este ponto quando o filtro exato não pode ser aplicado, ou seja, se existe pelo menos uma repetição de valor dentre as centralidades de cada autovetor associados aos índices dos grafos de entrada. Neste caso, uma árvore de soluções é construída de acordo com o algoritmo exato de *backtracking* apresentado em Lee (2007). Encontrada uma solução na descida da árvore, para verificar se esta define ou não um isomorfismo entre os grafos, utilizamos um teorema (aqui denotado por teorema PIG-PQA) que foi desenvolvido neste mesmo trabalho. Este teorema é a consolidação da reformulação do Problema de Isomorfismo de Grafos como um Problema Quadrático de Alocação (PQA, Loiola et al. (2007)). Para isso, deve-se construir um grafo completo valorado G' associado a um grafo G, cujas arestas recebem valor 1 se correspondem as arestas de G e 0 se seus vértices extremos correspondem a vértices não adjacentes em G. O teorema mencionado garante que o valor da solução do PQA relativo aos grafos G'_1 e G'_2 deve ser igual ao número de arestas de G_1 e G_2 para que estes sejam isomorfos.

O algoritmo proposto neste trabalho define os blocos de vértices com centralidades proporcionais para conduzir o caminhamento na árvore de busca, associando os vértices de um mesmo bloco, que possuam o mesmo grau e que gerem apenas associações entre arestas de G'_1 e G'_2 com mesmo peso. Caso esta associação não seja atendida, o algoritmo termina a exploração daquele ramo da árvore e realiza *backtracking* para um nível acima, continuando a exploração a partir deste ponto.

Portanto, a cada nível da árvore de busca gerada, uma nova associação de vértices é inserida na solução. Assim, a árvore possui profundidade igual ao número de vértices dos grafos. Deste modo, quando um nó folha da árvore de busca é atingido, um isomorfismo entre os grafos é identificado, finalizando a execução do algoritmo. Caso contrário, quando a exploração da árvore termina no seu nó raiz, ou seja, quando todas as possibilidades de associação entre os vértices foram testadas, porém sem sucesso, o algoritmo conclui que os grafos de entrada não são isomorfos. Para ratificar a solução encontrada, utilizamos o teorema supracitado.

O algoritmo proposto neste trabalho, o qual denotamos por AEPIG (Algoritmo Espectral para o Problema de Isomorfismo de Grafos), é formado pelas três fases explicadas acima. Seu pseudocódigo é apresentado em Algoritmo 1.

Algoritmo 1: **Algoritmo Espectral para o Problema de Isomorfismo de Grafos (AEPIG)**

Entrada: As matrizes de adjacência dos grafos G_1 e G_2
Saída: Sim (se os grafos são isomorfos) ou Não (caso contrário)
Calcular os índices λ_1 e λ_2 de G_1 e G_2, respectivamente `// Fase 1`
Calcular os autovetores positivos $\vec{x}^1$ e $\vec{x}^2$ associados, respectivamente, à λ_1 e λ_2
Ordenar de maneira crescente os componentes dos autovetores
se *($\vec{x}^1 \neq k\vec{x}^2$, $k \in \mathcal{R}^*$)* **então**
    G_1 e G_2 não são isomorfos
senão
    se *($\vec{x}^i = (x_1^i, ..., x_n^i)$, tal que $x_j^i \neq x_k^i$, $j, k = 1, \ldots, n$, $j \neq k$ e $i = 1, 2$)* `// Fase 2`
    então
        G_1 e G_2 são isomorfos
    senão
        Calcular $G_1^{'}$ e $G_2^{'}$ a partir de G_1 e G_2, respectivamente `// Fase 3`
        Realizar a descida na árvore de busca
        se *(o número de associações realizadas for igual a n (o teorema PIG-PQA é satisfeito))* **então**
            G_1 e G_2 são isomorfos
        senão
            G_1 e G_2 não são isomorfos
        fim se
    fim se
fim se

6 Resultados Computacionais

A fim de avaliarmos o desempenho do algoritmo AEPIG descrito na seção 5, realizamos a comparação do tempo de processamento deste com os algoritmos exatos propostos por Lee (2007) e Dharwadker & Tevet (2009). Para isto, utilizamos parte da base de dados da biblioteca VFLib (Santo et al. (2003)). A base de dados utilizada nos testes é formada por 3.000 pares de grafos conexos isomorfos gerados aleatoriamente, sendo divididos em três grupos de densidade $\eta = 0,01$, $\eta = 0,05$ e $\eta = 0,1$. Cada grupo possui 100 pares de grafos (que chamamos de instâncias) de tamanhos 20, 40, 60, 80, 100, 200, 400, 600, 800 e 1.000 vértices, com 1.000 pares de grafos em cada grupo. Para analisar os resultados, nomeamos cada grupo de grafos de acordo com a sua densidade de arestas, ficando $r001$ para grafos com $\eta = 0,01$, $r005$ para os de $\eta = 0,05$ e $r01$ para os de $\eta = 0,1$. Dividimos também cada grupo em conjuntos de instâncias de acordo com o número de vértices dos grafos, obtendo 10 conjuntos (20, 40, 60, ..., 1.000) em cada grupo de densidade. Na geração dos grafos da base de dados foi considerada a probabilidade η de uma aresta conectar dois vértices distintos, sendo assumida como uniforme e independente dos vértices. O número de arestas de cada grafo é dado por $\eta.n.(n-1)$, onde n é o número total de vértices do grafo. Porém, se este número não for suficiente para obter um grafo conexo, mais arestas são devidamente inseridas até a geração de um grafo conexo.

O algoritmo proposto neste trabalho foi implementado na Linguagem C, com a utilização da biblioteca time.h para a obtenção dos tempos de execução. Para o cálculo dos autovalores e autovetores foi utilizado o pacote CLAPACK versão 3.2.1, que é um pacote desenvolvido na Linguagem C que provê funções para cálculos de sistemas lineares. Os testes foram realizados em um computador com processador Intel® Core™ 2 Duo E7500 de 2.93GHz, 3GB de memória RAM, Sistema Operacional Linux Ubuntu 9.10 e kernel 2.6.31-20.

# Instâncias	AEPIG	Lee	DT
20	0,000170	0,000582	0,761822
40	0,000449	0,001569	8,008165
60	0,000896	0,001143	30,953340
80	0,000998	0,002599	81,035536
100	0,001665	0,000096	170,121170
200	0,010055	0,000424	2103,660660
400	0,070377	0,010047	–
600	0,229744	0,027049	–
800	0,591548	0,055887	–
1000	1,402799	0,094803	–

Tabela 1: Tempo médio (em segundos) dos algoritmos para os conjuntos de instâncias do grupo $r01$

# Instâncias	AEPIG	Lee	DT
20	0,000131	0,064017	0,726549
40	0,000325	5,389129	7,619884
60	0,000529	27,843959	29,599965
80	0,000998	42,007933	29,599965
100	0,001664	12,298832	179,678340
200	0,010090	1,299386	2340,182519
400	0,070791	0,202869	–
600	0,230232	0,044740	–
800	0,594165	0,077939	–
1000	1,380830	0,121431	–

Tabela 2: Tempo médio (em segundos) dos algoritmos para os conjuntos de instâncias do grupo $r005$

6.1 Análise dos Resultados

As Tabelas 1, 2 e 3 apresentam o tempo médio de execução para os conjuntos de instâncias dos grupos $r01$, $r005$ e $r001$, respectivamente. O tempo máximo de execução considerado foi de uma hora, sendo cancelada a execução do algoritmo que ultrapassasse este limite. A primeira coluna destas tabelas indica a ordem dos grafos e as demais, indicam o tempo médio em segundos de execução dos algoritmos AEPIG, Lee e DT, que denotam, respectivamente, o algoritmo proposto neste trabalho e os propostos em Lee (2007) e Dharwadker & Tevet (2009). O símbolo "–" na última coluna da tabela indica que a execução foi cancelada por ultrapassar o tempo máximo de execução estipulado.

A partir da Tabela 1, é possível perceber que o algoritmo AEPIG possui tempo médio de execução menor que Lee até o grupo de grafos de 80 vértices e em todos os grupos de grafos, se comparado com DT. Este último teve suas execuções canceladas por limite de tempo, a partir do grupo de grafos de 400 vértices.

A Tabela 2 mostra que para o grupo de grafos testados, tanto o AEPIG quanto Lee obtiveram em tempo viável a resposta para o problema, enquanto o DT obteve essa resposta para grafos com ordem até 200 vértices. Podemos observar que o algoritmo proposto neste trabalho foi bem mais rápido do que Lee para grafos com até 400 vértices. Já para as instâncias de ordem superior a 400, o melhor comportamento foi o do algoritmo Lee.

As médias dos tempos de execução dos algoritmos para cada conjunto de instâncias do grupo $r001$ são apresentadas na Tabela 3. Nela observamos que o algoritmo AEPIG é, na média, mais rápido que os outros dois e que executou em tempo computacional viável para todas as instâncias,

# Instâncias	AEPIG	Lee	DT
20	0,000120	51,878369	0,665118
40	0,000291	–	6,912850
60	0,000579	–	29,173299
80	0,001030	–	83,335449
100	0,001751	–	185,153810
200	0,010102	–	–
400	0,070463	–	–
600	0,231004	–	–
800	0,596862	–	–
1000	1,369227	–	–

Tabela 3: Tempo médio (em segundos) dos algoritmos para os conjuntos de instâncias do grupo $r001$

ao contrário dos demais. O algoritmo proposto por Lee e o algoritmo DT tiveram suas execuções canceladas por limite de tempo, a partir do grupo de grafos de 40 vértices e de 200 vértices, respectivamente. Para um par de grafos deste grupo, o algoritmo DT apresentou um resultado incorreto, indicando como não isomorfa a instância iso_r001_s20_06 (densidade $0,01$ - 20 vértices - instância 6), mostrando assim que ele não é necessário e suficiente para o isomorfismo entre dois grafos.

Analisando os resultados apresentados, verificamos que o algoritmo AEPIG tem tempo de processamento menor do que o algoritmo DT em todos os conjuntos de instâncias testadas, para todos os valores de densidade. Além disso, comparando-o com o algoritmo Lee, seu tempo de execução torna-se menor à medida que a densidade de arestas dos grafos diminui. Isso pode ser explicado pelo fato do AEPIG gerar blocos de vértices de centralidades proporcionais a fim de conduzir eficientemente a descida na árvore de busca (descrita na subseção 5.3), reduzindo assim o espaço de soluções do PIG. Com a mesma finalidade, o algoritmo desenvolvido por Lee gera blocos de vértices de mesmo grau. Com isso, em ambos os algoritmos, um vértice de um grafo somente pode ser associado a um vértice de outro grafo se ambos estiverem em um mesmo bloco (respectivamente, de centralidade e de grau).

Desta forma, o número de blocos gerados influencia a complexidade da busca pela solução, pois quanto maior é este número, menor será o espaço de soluções viáveis do problema, melhorando os tempos de processamento dos algoritmos. Observando os resultados apresentados pela Tabela 4, verificamos que o algoritmo AEPIG gera mais blocos de centralidade do que o algoritmo Lee gera blocos de graus, tendo assim a descida na árvore de busca mais eficiente, caso seja necessária a sua utilização. Em média, nos testes para todos os conjuntos de densidades, o número de blocos de centralidade distintas foi bem maior que o de graus distintos. Pode-se observar que, para $r001$, o número de blocos de graus distintos é bastante inferior, o que explica o tempo inviável (ou seja, superior ao limite de tempo estipulado) para o algoritmo Lee neste conjunto de instâncias.

7 Conclusão e Trabalhos Futuros

Neste trabalho investigamos a utilização de conceitos da TEG para auxiliar a construção de algoritmos para detecção de isomorfismo de grafos. Dois resultados teóricos que consideram informações do espectro e das centralidades dos vértices dos grafos foram apresentados. Além deles, o algoritmo AEPIG para detecção de isomorfismo de grafos baseado nestes resultados, munido de dois filtros, um exato e outro heurístico, é proposto.

De acordo com todos os testes realizados, observamos que um gargalo computacional do algoritmo AEPIG é a função para cálculo dos autovalores e autovetores dos grafos. A função utilizada,

	$r01$		$r005$		$r001$	
# Instâncias	BC	BG	BC	BG	BC	BG
20	20	7	19	5	16	4
40	40	10	40	8	34	6
60	60	14	60	11	55	7
80	80	16	80	13	75	7
100	100	18	100	14	97	8
200	200	27	200	22	200	11
400	400	41	400	33	400	17
600	600	52	600	41	600	21
800	800	62	800	48	800	25
1000	1000	70	1000	55	800	25

Tabela 4: Número médio de blocos de centralidades (BC) do AEPIG e de graus (BG) do algoritmo Lee

oriunda da biblioteca CLAPACK, contribuiu com uma parcela considerável no tempo total de processamento do algoritmo, sendo responsável, em média, por 90% deste tempo, uma vez que na maioria dos testes, a Fase 3 não foi executada, ou seja, a árvore de busca de soluções não precisou ser gerada. Mais precisamente, dos 1.000 testes realizados com o AEPIG para o conjunto de instâncias $r01$, em apenas $0,7\%$ delas, a árvore de busca foi necessária para encontrar o isomorfismo; para o conjunto $r005$, apenas $7,1\%$ e, finalmente, no último conjunto ($r001$) foi necessária a geração da árvore para um número maior de instâncias, $52,3\%$. Desta forma, o Teorema 1 se mostrou poderoso na detecção do isomorfismo para a grande maioria dos testes realizados. Nos outros casos, a exploração da árvore de busca de soluções guiada pelos blocos de centralidades se mostrou bastante eficiente.

Como trabalhos futuros, pretendemos investigar na literatura funções mais eficientes para o cálculo de autovalores e autovetores, além de comparar esse algoritmo com outros bastante citados na literatura, como por exemplo o algoritmo *Nauty*.

Agradecemos à FAPES (Fundação de Amparo à Pesquisa do Espírito Santo) pela concessão da bolsa de mestrado para a realização deste trabalho.

Referências

Abreu, N. M. M. (2005). Teoria espectral dos grafos: um híbrido entre a álgebra linear e a matemática discreta e combinatória com origens na química quântica. *Tendências em Matemática Aplicada e Computacional*, 6(1):1–10.

Abreu, N. M. M., Del-Vecchio, R. R., Vinagre, C. T. M., & Stevanović, D. (2007). Introdução à teoria espectral de grafos com aplicações. In *Notas em Matemática Aplicada*. SBMAC.

Arvind, V. & Torán, J. (2005). Isomorphism testing: Perspective and open problems. *Bulletin European Association of Theoretical Computer Science*, 86:66–84.

Berge, C. (1991). *Graphs*. Elsevier, Amsterdam, 3ª edition.

Bonacich, P. (2007). Some unique properties of eigenvector centrality. *Social Networks*, 29(4):555–564.

Cordella, L. P., Foggia, P., Sansone, C., & Vento, M. (1999). Performance evaluating of the vf graph matching algorithm. In *Proc. of the 10th International Conference on Image Analysis and Processing*, pages 1172–1177, Washington, DC, USA. IEEE Computer Society Press.

Cordella, L. P., Foggia, P., Sansone, C., & Vento, M. (2001). An improved algorithm for matching large graphs. In *In: 3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pages 149–159, Ischia, Italy.

Dalcumune, E. (2008). Algoritmos quânticos para o problema do isomorfismo de grafos. Dissertação de Mestrado, Laboratório Nacional de Computação Científica, Petrópolis.

Dharwadker, A. & Tevet, J. (2009). The graph isomorphism algorithm. In *Proceedings of the Structure Semiotics Research Group*, Estônia. Eurouniversity Tallinn.

Ferreira Jr, A., Pereira, A. S., & Carreira, T. G. M. (2001). *Proposta de Implementação de Sistema de Segurança Utilizando Sistemas Biométricos*. Universidade da Amazônia, Centro de Ciências Exatas e Tecnológicas, Belém. Trabalho de Conclusão de Curso.

Fortin, S. (1996). The graph isomorphism problem. Technical report, University of Alberta, Edmonton, Alberta, Canada.

Grassi, R., Stefani, S., & Torriero, A. (2007). Some new results on the eigenvector centrality. *Journal of Mathematical Sociology*, 31(3):237–248.

Hogben, L. (2009). Spectral graph theory and the inverse eigenvalue problem of a graph. *Chamchuri Journal of Mathematics*, 1(1):51–72.

Jenner, B., McKenzie, J. K. P., & Torán, J. (2003). Completeness results for graph isomorphism. *Journal of Computer and System Sciences*, 66(3):549–566.

Lee, L. (2007). Reformulação do problema de isomorfismo de grafos como um problema quadrático de alocação. Dissertação de Mestrado, Universidade Federal do Espírito Santo - Centro Tecnológico - Departamento de Informática, Vitória.

Loiola, E. M., Abreu, N. M. M., Netto, P. O. B., Hahn, P., & Querido, T. M. (2007). A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657–690.

McKay, B. D. (1981). Practical graph isomorphism. In *Congressus Numerantium*, volume 30, pages 45–87.

Oliveira, M. O. & Greve, F. G. (2005). Um novo algoritmo de refinamento para testes de isomorfismo em grafos. In *XXV CSBC*, pages 140–148, São Leopoldo-RS. SBC.

Santo, M. D., Foggia, P., Sansone, C., & Vento, M. (2003). A large database of graphs and its use for benchmarking graph isomorphism algorithms. *Pattern Recogn. Letters*, 24(8):1067–1079.

Sorlin, S. & Solnon, C. (2004). A global constraint for graph isomorphism problems. In *CPAIOR, Lecture Notes in Computer Science*, volume 3011, pages 287–302. Springer.

Uehara, R., Toda, S., & Nagoya, T. (2005). Graph isomorphism completeness for chordal bipartite graphs and strongly chordal graphs. *Discrete Applied Mathematics*, 145(3):479–482.

Ullmann, J. (1976). An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23(1):31–42.

Xiutang, G. & Kai, Z. (2008). Simulated annealing algorithm for detecting graph isomorphism. *Journal of Systems Engineering and Electronics*, 19(4):52–57.

Zager, L. (2005). Graph similarity and matching. Dissertação de Mestrado, Department of Electrical Engineering and Computer Science. Massachusetts Institute of Technology.