O Uso de Autocentralidades na Resolução do Problema de Isomorfismo de Grafos Regulares[1]

Diego Barcelos Rodrigues
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
diego.barcelos@gmail.com, crangel@inf.ufes.br, boeres@inf.ufes.br

RESUMO

A Teoria Espectral de Grafos (TEG) busca analisar propriedades dos grafos através de matrizes representativas de grafos e seus espectros. De uma propriedade proveniente da TEG, a autocentralidade, surge um importante invariante para o Problema de Isomorfismo de Grafos: se dois grafos são isomorfos então eles possuem autocentralidades proporcionais. Porém, esta propriedade não pode ser usada diretamente para resolução do Problema de Isomorfismo de Grafos Regulares (PIGR), pois todo grafo regular possui autocentralidades iguais. Este trabalho apresenta uma estratégia para resolver o PIGR através do uso das autocentralidades para podar a árvore de busca e restringir as possibilidades de mapeamento.

PALAVRAS CHAVE. Isomorfismo. Espectro. Autocentralidade.

Área principal (Teoria e Algoritmos em Grafos).

ABSTRACT

Spectral Graph Theory (SGT) studies graph properties by graph representation matrix and its spectrum. A property from SGT, the eigencentrality, provides an important invariant to Graph Isomorphism Problem: if two graphs are isomorphic, they have proportional eigencentralities. However, this property can not be directly used for solving the Regular Graph Isomorphism Problem (RGIP), as every regular graph has the same eigencentralities. This paper presents a strategy for solving the RGIP through the use of eigencentralities to prune the search tree and restricting the possibilities for mapping.

KEYWORDS. Isomorphism. Spectro. Eigencentrality.

Main area (Algorithms and Graph Theory).

[1]Este trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Espírito Santo (FAPES).

1. Introdução

O Problema de Isomorfismo de Grafos (PIG) é um problema de grande interesse entre os pesquisadores, pois é sabido que o PIG pertence à classe NP (Presa & Anta 2009), porém, continua sendo chamado de indefinido quando consideradas as classes de complexidade P e NP-completo (Porumbel (2009), Presa & Anta (2009)). A existência de alguns algoritmos polinomiais para algumas classes de grafos, por exemplo, grafos de valência limitada (Luks (1982)), levam alguns pesquisadores a acreditarem na existência de um algoritmo polinomial que comprove que o PIG está em P. Porém, existem algumas classes de grafos que exigem um comportamento exponencial dos mais eficientes algoritmos (Miyazaki (1996)), levando outros pesquisadores a acreditarem que o PIG não está em P. Ainda existem aqueles que acreditam que esse problema não seja pertencente a classe P e nem a classe NP-completa como Karp (1972) e Oliveira & Greve (2005).

Todavia, vários algoritmos eficientes foram desenvolvidos para o PIG como, por exemplo, o algoritmo denominado VF (Cordella et al. (1999)) que consiste em uma busca em profundidade com técnicas de poda, a sua versão melhorada denominada VF2 (Cordella et al. (2001)) e o filtro paramétrico IDL(d) (Sorlin & Solnon (2008)), que faz uso de um forte invariante, distâncias entre os vértices, para rerrotular grafos e restringir as possibilidades de mapeamento. Os algoritmos *Conauto* (Presa & Anta (2009)) e *Nauty* (McKay (1981)) buscam identificar o isomorfismo entre os grafos através de rotulação canônica e automorfismos.

Outro fator que atrai os pesquisadores são as aplicações do PIG, que pode ser usado para modelar problemas em diversas áreas, por exemplo, na identificação de compostos químicos em Química (Corniel & Gotlieb (1970)), reconhecimento de moléculas de proteína em Biologia (Abdulrahim & Misra (1998)), reconhecimento biométrico para identificação de pessoas (Nandi (2006)), entre outros.

Dois grafos são isomorfos se existir uma função bijetora que mapeie os conjuntos de vértices preservando as suas relações de adjacências. O PIG consiste em verificar se dois grafos são isomorfos ou não, apresentando a bijeção mencionada, no caso afirmativo.

A principal estratégia utilizada na resolução do PIG é a identificação de vértices similares para mapeamento, isto é, vértices que possuam as mesmas características extraídas de propriedades estruturais dos grafos como sequências de graus, distâncias entre os vértices, centralidades de autovetor, entre outras.

A Teoria Espectral de Grafos (TEG) busca analisar propriedades dos grafos através de suas matrizes representativas (Adjacência, Laplaciana, Laplaciana Sem Sinal, entre outras) e seus espectros. Alguns pesquisadores utilizam TEG como ferramenta para identificar características comuns entre os vértices do par de grafos comparado (Santos et al. (2010) e Rajasekaran & Kundeti (2009)). Através dessas características é possível confirmar ou refutar a presença de isomorfismo, ou simplesmente reduzir o espaço de busca no qual são feitas tentativas de mapeamento.

Grafos regulares constituem a classe de grafos de maior desafio, principalmente os fortemente regulares, pois são grafos que possuem vértices difíceis de serem distinguidos uns dos outros, o que torna mais difícil identificar se grafos desse tipo são isomorfos, motivando a busca por um algoritmo eficiente para resolução do PIGR. Além disso, não foi encontrada uma abordagem que utilize autocentralidades como base de um procedimento de resolução deste problema. A principal razão para isto é que as informações advindas do vetor de autocentralidades não possibilitam distinção alguma dos vértices de grafos regulares, pois o vetor de autocentralidades desses tipos de grafos é proporcional ao vetor unitário (Grassi et al. (2007)). Este trabalho apresenta uma abordagem de resolução do Problema de Isomorfismo de Grafos Regulares (PIGR) que utiliza conceitos de TEG, as autocentralidades, para buscar um mapeamento que atenda às condições de isomorfismo.

A próxima seção trata do Problema de Isomorfismo de Grafos apresentando sua definição

formal e exemplos de grafos isomorfos e não isomorfos. Na Seção 3 são apresentadas algumas definições e resultados da literatura referente à Teoria Espectral de Grafos. Na Seção 4 o algoritmo proposto neste trabalho é apresentado e sua consistência é avaliada na Seção 5. Por fim, na Seção 6 o trabalho é concluído e perspectivas futuras são apresentadas.

2. O Problema de Isomorfismo de Grafos

Neste problema é de interesse averiguar a existência de uma correspondência biunívoca definida sobre os conjuntos de vértices de dois grafos que preserve as suas adjacências. Mais formalmente:

Dois grafos G_1 e G_2 são isomorfos se, e somente se, existe uma função bijetora $\phi : V(G_1) \rightarrow V(G_2)$ tal que, $\forall u, v \in V(G_1)$, $\{u, v\} \in E(G_1) \Leftrightarrow \{\phi(u), \phi(v)\} \in E(G_2)$ (Diestel (2006)).

Exemplos de pares de grafos isomorfos e não isomorfos podem ser vistos na Figura 1. É fácil ver que os grafos G_1 e G_2 são isomorfos, apresentando a função bijetora $\phi : V(G_1) \rightarrow V(G_2)$ com a qual se obtém o mapeamento $\{(1,6),(2,3),(3,2),(4,1),(5,4),(6,5)\}$. Porém, não é possível apresentar uma função bijetora $\psi : V(G_1) \rightarrow V(G_3)$ (ou $\psi : V(G_2) \rightarrow V(G_3)$) com a qual seja obtido um mapeamento que atenda à definição de isomorfismo, pois esses grafos possuem claramente diferenças estruturais: G_1 e G_2 são planares enquanto G_3 não é.

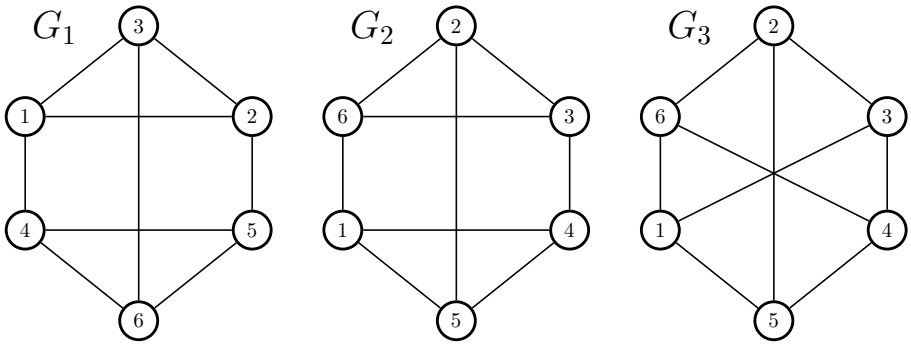

Figura 1: Grafos isomorfos (G_1 e G_2); grafos não isomorfos (G_1 e G_3; G_2 e G_3).

3. Teoria Espectral de Grafos

O espectro de um grafo, definido a partir do seu conjunto de autovalores, depende da matriz de representação utilizada. A Teoria Espectral de Grafos busca relacionar propriedades dos grafos através de suas matrizes de representação e seus espectros. Sendo assim, é possível utilizar teoria espectral de grafos para identificar características que devem ser comuns a grafos isomorfos, no intuito de eliminar a possibilidade de isomorfismo, filtrar possibilidades de mapeamento entre os conjuntos de vértices dos grafos ou até mesmo apresentar uma bijeção entre os vértices desses conjuntos.

A matriz de representação de grafos utilizada neste trabalho é a matriz de adjacência. Na sequência são apresentadas algumas definições importantes relacionadas a essa matriz referente a grafos simples, não direcionados e conexos.

Seja G um grafo com n vértices. A **matriz de adjacência** $A(G)$ de G é a matriz quadrada de ordem n cujas entradas são definidas por:

$$a_{ij} = \begin{cases} 1, & se\ \{v_i, v_j\} \in E(G),\ v_i, v_j \in V(G); \\ 0, & caso\ contrário. \end{cases}$$

O polinômio característico da matriz de adjacência $A(G)$ do grafo G é dado por $det(\lambda I - A(G))$ e é denominado **polinômio característico de** G, denotado por $p_G(\lambda)$.

Seja λ uma raiz do polinômio característico de G, então λ é denominado um **autovalor de** G quando é uma raiz de p_G (λ). Se $A(G)$ possui s autovalores distintos $\lambda_1 > \ldots > \lambda_s$ com multiplicidades m(λ_1), ..., m(λ_s), respectivamente, o **espectro de** G, denotado *spect(G)*, é definido como a matriz 2 $\times$ s, onde na primeira linha se encontra os *s* autovalores distintos de *G* e na segunda linha suas respectivas multiplicidades. Ou seja:

$$spect(G) = \begin{bmatrix} \lambda_1 & \ldots & \lambda_s \\ m(\lambda_1) & \ldots & m(\lambda_s) \end{bmatrix}.$$

O maior autovalor de G é denominado **índice de** G e denotado por λ_1. O espectro é a base da Teoria Espectral de Grafos, pois é dele que os pesquisadores buscam obter informações relevantes para resolução dos mais variados tipos de problema como Corte Maximal, PIG, Problema de Determinação de Energia Molecular, entre outros (Abreu et al. (2007)).

Sabe-se que o espectro é um invariante quando se trata de isomorfismo de grafos, portanto, se dois grafos são isomorfos eles possuem o mesmo espectro (Abreu et al. (2007)). Porém, a recíproca desta afirmação não é verdadeira. A Figura 2 ilustra um par de grafos que possuem o mesmo espectro mas não são isomorfos.

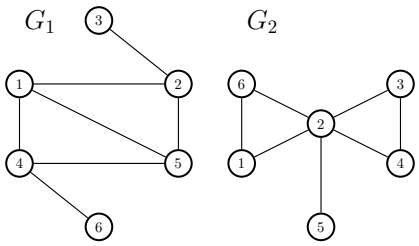

Figura 2: Grafos não isomorfos de mesmo espectro.

Um **autovetor** não negativo associado ao índice de um grafo G é um vetor não nulo v, tal que $M(G)v = \lambda v$, onde $M(G)$ é uma matriz que representa G e λ é um autovalor dessa matriz. O vetor v é dito autovetor associado ao autovalor λ. O autovetor não negativo associado ao índice de um grafo é também conhecido como **autovetor principal**.

Seja x o autovetor principal de um grafo G simples e conexo. A **centralidade de autovetor** (autocentralidade) x_i de um vértice de índice i em $V(G)$ é definida como a i-ésima coordenada do autovetor principal x. Decorre desta definição que, se dois grafos são isomorfos eles possuem autocentralidades proporcionais (Santos (2010)).

Uma limitação de abordagens espectrais a ser destacada neste trabalho diz respeito à obtenção do espectro e das autocentralidades dos grafos. Estes valores são tipicamente calculados de forma aproximada (Spielman (2007)). Quando usados na identificação de isomorfismo entre dois grafos, pode levar a conclusões erradas e/ou resultar em algoritmos mais lentos. Além disso, não é possível utilizar as autocentralidades para resolução do PIGR como foi feito em Santos et al. (2010), pois grafos regulares possuem autocentralidades iguais (Grassi et al. (2007)).

Na próxima seção é apresentada a proposta deste trabalho que é voltada para a resolução do PIGR através do uso das autocentralidades.

4. Proposta de um algoritmo para o PIGR baseado no uso de autocentralidades

Dados dois grafos regulares simples e conexos o algoritmo proposto neste trabalho, denominado *Autocentralidades na Detecção de Isomorfismo de Grafos Regulares* (ADIGR), busca evidenciar assimetrias nesses grafos através da inclusão de vértices e arestas, de forma a facilitar a busca por um mapeamento que atenda às condições de isomorfismo (Rajasekaran & Kundeti (2009), Santos et al. (2010)). O algoritmo é baseado na ideia de que em caso de isomorfismo entre dois grafos G_1 e G_2, ao ser modificada uma característica em G_1 e *modificação equivalente*[2] for realizada em G_2, os grafos resultantes dessas modificações também serão isomorfos.

A Figura 3 apresenta um exemplo de uma modificação equivalente, onde G'_1 é o grafo resultante das adições do vértice 7 e da aresta $\{2, 7\}$ ao grafo G_1 da Figura 1, e G'_2 é o grafo resultante de uma modificação equivalente caracterizada pelas adições do vértice 7 e da aresta $\{3, 7\}$ a G_2, também da Figura 1.

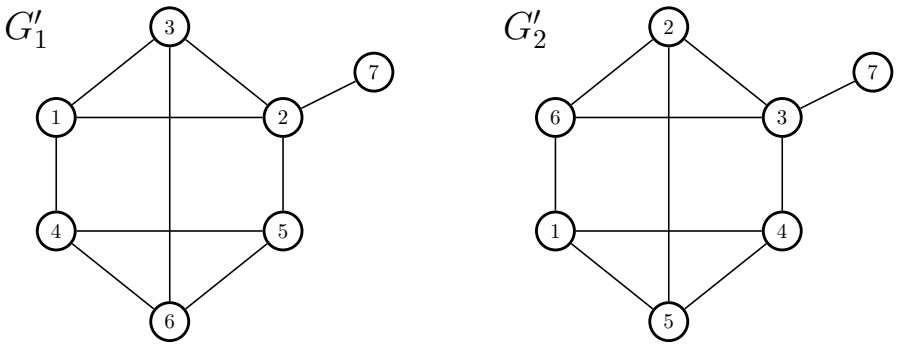

Figura 3: *Exemplo da aplicação de uma modificação equivalente aos grafos G_1 e G_2 da Figura 1.*

De posse de dois grafos k-regulares G_1 e G_2 simples e conexos de ordem n, o ADIGR realiza n modificações equivalentes em G_1 e G_2 de maneira que os respectivos grafos resultantes, G_1^n e G_2^n, sejam simples e conexos. Em cada modificação realizada, cada vértice adicionado a um dos grafos é ligado somente a um **único** vértice do mesmo. Para cada $v \in V(G_1)$ que tem seu grau e autocentralidade modificados, o algoritmo busca $u \in V(G_2)$ que, após sofrer modificação equivalente à v, possua os mesmos valores de grau e autocentralidade. Dessa forma é esperado em caso de isomorfismo entre G_1 e G_2 que, para cada sequência de adições de vértices e arestas realizadas em G_1, exista uma sequência de modificações equivalentes a serem realizadas em G_2 que permitam a identificação de um mapeamento atendendo às condições de isomorfismo.

O ADIGR é apresentado no Algoritmo 1. É fácil perceber que o algoritmo realiza uma busca em profundidade com poda na árvore relativa a todos os mapeamentos que podem ser realizados entre $V(G_1)$ e $V(G_2)$, na tentativa de encontrar um que caracterize o isomorfismo entre os grafos comparados. A poda da árvore de busca é realizada através da utilização das autocentralidades dos vértices, antes e depois de cada modificação equivalente, e do autovetor principal. Além disso, o algoritmo faz uso da técnica do *backtracking*, caso não seja possível encontrar $u \in V(G_2)$ que (após sofrer modificação equivalente a $v \in V(G_1)$) fique com o mesmo valor de autocentralidade e os grafos resultantes com vetores principais proporcionais.

[2]Neste trabalho é considerado como modificação equivalente a aplicação do mesmo conjunto de operações de edição de grafos (inserção de vértice e de aresta) em ambos os grafos.

As funções $ac(\cdot)$ e $avp(\cdot)$ representam, respectivamente, a autocentralidade de um vértice e o autovetor principal de um grafo. O algoritmo utiliza os seguintes critérios:

- **C1 - Seleção do vértice a ser modificado**

  Estabelece que seja selecionado um vértice de grau *k* que não seja vizinho de vértices adicionados ao grafo, ou seja, $v \in V(G_1) \subseteq V(G_1^i)$ tal que, $\forall\ w \in V(G_1^i) \backslash V(G_1)$, $\{v, w\} \notin E(G_1^i) \backslash E(G_1)$, $0 \leq i \leq n$, onde G_1^i é o grafo resultante de i modificações realizadas em G_1. Este critério é aplicado somente a vértices de G_1.

- **C2 - Seleção de um candidato a mapeamento**

  Este critério é utilizado para reduzir as tentativas de mapeamento. Ele estabelece que seja selecionado para tentativa de mapeamento com $v \in V(G_1) \subset V(G_1^i)$, um vértice $u \in V(G_2) \subseteq V(G_2^{i-1})$, $0 \leq i \leq n$, tal que grau(u) = k, a autocentralidade de u seja igual a x, onde x é a autocentralidade de $v \in V(G_1) \subseteq V(G_1^{i-1})$, e u ainda não tenha sido mapeado com v.

- **C3 - Critério de mapeamento**

  Determina que os vértices selecionados em C1 e C2 sejam mapeados somente se possuírem as mesmas autocentralidades e os grafos resultantes $V(G_1^i)$ e $V(G_2^i)$, $1 \leq i \leq n$, possuírem autovetores principais proporcionais. Além disso, este critério desempenha papel fundamental na poda da árvore de busca, pois caso nenhuma das tentativas de mapeamento atenda a este critério, o ADIGR realiza *backtracking*.

Uma vez selecionado o vértice $v \in V(G_1) \subset V(G_1^{i-1})$, de acordo com o critério $C1$, novos vértices são ligados a ele e o autovetor principal do grafo resultante G_1^i é calculado (*Passo 3*). Em seguida o algoritmo procura um vértice $u \in V(G_2) \subset V(G_2^{i-1})$ que atenda ao critério $C2$ (*Passo 4* - linhas 12 e 13). Se não houver tal vértice e não for possível realizar *backtracking* (*Passo 5*), o algoritmo identifica os grafos como não isomorfos (*Passo 7*). Se não houver tal vértice e $i \neq 0$, o algoritmo realiza $backtracking$ (*Passo 5*). Se tal u for encontrado, i novos vértices são ligados a u e o autovetor principal do grafo resultante G_2^i é calculado (*Passo 4* - linhas 14 e 15). Então, se o critério $C3$ for atendido, v e u são mapeados (*Passo 4* - linha 17) e o algoritmo desce ao próximo nível da árvore (*Passo 6*). Caso contrário, o algoritmo remove os novos vértices ligados a u (*Passo 4* - linha 18) e busca em $V(G_2) \subset V(G_2^{i-1})$ outro candidato a mapeamento com v. Por fim, caso um vértice folha seja alcançado, isto é, caso G_1^n e G_2^n sejam gerados, o algoritmo retorna o mapeamento encontrado.

5. Resultados Computacionais

Para testar o ADIGR foram utilizados grafos regulares gerados aleatoriamente com um programa de geração aleatória de grafos regulares implementado por Luchi & Rangel (2010). Trata-se da implementação de um algoritmo de tentativa e erro que gera grafos *k*-regulares de ordem n a partir do grafo 2-regular de mesma ordem, no qual são realizadas n^2 tentativas de inserções aleatórias de arestas sem que o grau de qualquer dos vértices torne-se maior que k. Ao final dessas tentativas é feita uma busca por pares de vértices que tenham graus inferiores a k para, então, continuar com o processo de inserção de arestas até que não existam mais pares de vértices {u,v} tais que os graus de u e v sejam inferiores a k. Neste ponto é verificado se o grafo gerado é regular, em caso afirmativo o grafo é armazenado em um arquivo, caso contrário, o processo de inserção aleatória de arestas é retomado a partir do grafo 2-regular.

Algoritmo 1: *Algoritmo ADIGR.*

Dados: $[Dois\ grafos\ G_1\ e\ G_2\ k\text{-}regulares\ simples\ de\ mesma\ ordem\ n]$

Passo *1* :
  $G_1^0 \leftarrow G_1$; $G_2^0 \leftarrow G_2$; $i \leftarrow 1$.
  $Calcule\ os\ autovetores\ principais\ de\ G_1^0\ e\ G_2^0$;

Passo *2* :
  Se $(\nexists\ \mathbf{v} \in V(G_1) \subseteq V(G_1^{i-1})\ de\ grau\ \mathbf{k})$ **então** $V\acute{a}\ ao\ Passo\ 7$;
  $Escolha\ \mathbf{v} \in V(G_1) \subseteq V(G_1^{i-1})\ de\ grau\ \mathbf{k}$.

Passo *3* :
  $\mathbf{x} \leftarrow$ ac**(v)**;
  $Obtenha\ G_1^i\ ligando\ i\ novos\ v\acute{e}rtices\ a\ \mathbf{v}$;
  $Calcule\ o\ autovetor\ principal\ de\ G_1^i$;

Passo *4* :
  Para cada $\boldsymbol{u} \in V(G_2) \subseteq V(G_2^{i-1})$ **faça**
    Se $((grau(\boldsymbol{u}) = \boldsymbol{k})\ \&\ (ac(\boldsymbol{u}) = \boldsymbol{x}))$, **então**
      $Obtenha\ G_2^i\ ligando\ i\ novos\ v\acute{e}rtices\ a\ \mathbf{u}$;
      $Calcule\ o\ autovetor\ principal\ de\ G_2^i$;
      Se $(avp(G_2^i)$ *é proporcional a* $avp(G_1^i)\ \&\ ac(\boldsymbol{u}) = ac(\boldsymbol{v}))$, **então**
        $Mapeie\ \mathbf{u}\ e\ \mathbf{v}\ e\ v\acute{a}\ ao\ Passo\ 6$;
      $Remova\ os\ novos\ v\acute{e}rtices\ que\ foram\ ligados\ a\ \mathbf{u}$;

Passo *5* $(Backtracking)$:
  $i \leftarrow i$ - 1;
  Se $(i < 1)$, **então** $V\acute{a}\ ao\ Passo\ 7$;
  $Remova\ os\ novos\ v\acute{e}rtices\ que\ foram\ ligados\ a\ \mathbf{v}$;
  $Recupere\ \mathbf{w} \in V(G_1^{i+1})\ que\ foi\ selecionado\ na\ modificação\ de\ número\ i$;
  $Remova\ os\ novos\ v\acute{e}rtices\ que\ foram\ ligados\ ao\ v\acute{e}rtice\ \mathbf{s}\ mapeado\ com\ \mathbf{w}$;
  $Desfaça\ o\ mapeamento\ entre\ \mathbf{s}\ e\ \mathbf{w}$; $\mathbf{v} \leftarrow \mathbf{w}$;
  $Calcule\ o\ autovetor\ principal\ de\ G_1^i$;
  $\mathbf{x} \leftarrow ac(\mathbf{v})$;
  $V\acute{a}\ ao\ Passo\ 4$.

Passo *6* :
  $i \leftarrow i$ + 1;
  Se $(i \leq n)$, **então** $V\acute{a}\ ao\ Passo\ 2$.

Passo *7* :
  Se $(i = n + 1)$, **então**
  "$Apresentar\ o\ mapeamento\ V(G_1) \rightarrow V(G_2)\ encontrado$."
  Senão "$Os\ grafos\ G_1\ e\ G_2\ não\ são\ isomorfos$!"

Foi gerado um grupo de grafos regulares de ordens 100, 150, 200, 250, 300, 350, 400 e 450, cada grafo com *densidade*[3] percentual de 4%, denotado por $grd04$. Para cada valor de n foram gerados 10 grafos com essas características. Grafos regulares de ordens 25, 50, 75 e 100 vértices e densidades percentuais nos intervalos [10,20), [20,30), [30,40) e [40,50) também foram gerados com a mesmo programa, para verificar o comportamento do algoritmo proposto com grafos não

[3]Neste trabalho é considerado como densidade de um grafo de ordem n e m arestas a relação $\frac{m}{t}$, onde t representa o máximo de arestas que um grafo de ordem n pode conter.

esparsos. Neste caso, para cada par $\{n, d\}$, onde n é o número de vértices e d a densidade, foram gerados 10 grafos. Esse segundo grupo de grafos foi denotado por $grd10$-49. As instâncias do problema foram formadas por pares de grafos distintos de mesma ordem e densidade. Isto resultou em 45 instâncias para cada par $\{n, d\}$. Além dessas instâncias, cinco pares de grafos regulares isomorfos de ordens e graus k iguais a, respectivamente, {10, 12, 17, 20, 30} e {3, 5, 5, 3, 3}, foram obtidos de Dharwadker & Tevet (2009) para testar o ADIGR com instâncias isomorfas.

Para calcular os vetores de autocentralidades, foi utilizada a função *ssyevr_* do pacote CLAPACK versão 3.2.1, que é um pacote obtido do Lapack (*Linear Algebra PACKage*) através da ferramenta de conversão *f2c* presente no próprio CLAPACK, que converte as rotinas de cálculo de sistemas lineares do LAPACK implementadas em Linguagem Fortran para a Linguagem C. A função *ssyevr_* é uma das mais eficientes para o cálculo de autovetores de matrizes simétricas do tipo ponto flutuante presente no CLAPACK. Essa função permite calcular apenas uma faixa de autovalores e seus autovetores associados. Neste trabalho, esta função foi utilizada de forma a retornar apenas o índice do grafo e seu autovetor associado (o autovetor principal).

Os testes computacionais foram realizados em um computador com processador Intel® Core™ 2 Duo T6600 de 2.2GHz, 2GB de memória RAM e Sistema Operacional Ubuntu 10.04 32 bits. Os tempos de execução, em segundos, obtidos através da função *gettimeofday* da biblioteca *sys/time.h* são apresentados nas Tabelas 1 e 2 relativas ao grupo $grd10$-49 e na Tabela 3, ao grupo $grd04$. Cada tabela apresenta os tempos médios, mínimos e máximos, obtidos com cada conjunto de 45 *instâncias de mesma ordem e densidade*[4]. As Tabelas 1 e 3 também apresentam, na coluna **ssyevr(%)**, a média dos percentuais relativos ao tempo gasto com as chamadas à função *ssyevr_*.

Sendo o objetivo dos testes computacionais avaliar a consistência do algoritmo proposto, foi utilizado o *Nauty* (McKay (1981)), um algoritmo exato dos mais citados e utilizados na literatura relativa ao PIG. Todas as instâncias formadas com os grafos dos grupos $grd04$ e $grd10$-49 foram classificadas como não isomorfas pelo *Nauty*, o mesmo ocorrendo com o ADIGR.

A Tabela 1 foi organizada de forma a possibilitar a observação dos tempos a medida que a densidade cresce e a ordem é mantida. A Tabela 2 foi organizada usando uma estratégia inversa: a medida que a ordem dos grafos cresce, a densidade é mantida.

A Tabela 2 mostra a influência da ordem de grafos não esparsos sobre o desempenho do ADIGR. Pelos resultados apresentados nesta tabela observa-se que os tempos aumentam a medida que a ordem das instâncias cresce, principalmente aquelas de ordem 100, que apresentam tempos significativamente mais elevados que as demais, exceto para o intervalo [10,20).

A Tabela 3 mostra o comportamento do ADIGR com instâncias formadas por grafos do grupo $grd04$.

A partir dos resultados apresentados nas Tabelas 1 e 2, observa-se que o tempo de processamento não é diretamente proporcional ao aumento da densidade dos grafos. A natureza das instâncias não justifica este comportamento pois os grafos não possuem características estruturais que possam ser destacas, uma vez que são gerados aleatoriamente. No entanto, o uso da função de precisão simples *ssyevr_* pode ter levado o algoritmo a seguir por caminhos inviáveis na árvore de soluções, aumentando o tempo de sua exploração. A partir da tabela 3 é possível observar melhor a influência da ordem dos grafos no desempenho do algoritmo proposto, pois neste caso, além de serem de maior ordem, existe também uma maior diversidade dos valores de n. Nota-se que o aumento do tempo de processamento acompanha a ordem dos grafos.

Santos et al. (2010) relataram que o cálculo dos espectros e autovetores principais, obtidos com a versão de precisão dupla da função *ssyevr_* do CLAPACK (*dsyevr_*), representou um

[4] A ordem e a densidade de uma instância corresponde a ordem e a densidade dos grafos que a compõe.

Ordem	Densidade(%)	Média(s)	Mínimo(s)	Máximo(s)	ssyevr(%)
25	[10,20)	0,002921	0,002406	0,005964	97,96
25	[20,30)	0,002799	0,002426	0,005080	97,94
25	[30,40)	0,002926	0,002435	0,008332	97,73
25	[40,50)	0,002921	0,002458	0,008152	98,25
50	[10,20)	0,013724	0,012572	0,016522	99,18
50	[20,30)	0,014696	0,012690	0,014696	98,80
50	[30,40)	0,013920	0,012808	0,024368	98,63
50	[40,50)	0,027170	0,012886	0,060788	98,92
75	[10,20)	0,045597	0,042210	0,045597	99,46
75	[20,30)	0,048721	0,042091	0,072284	99,28
75	[30,40)	0,573908	0,042770	0,907725	99,31
75	[40,50)	0,197122	0,068596	0,496697	99,45
100	[10,20)	0,111050	0,109281	0,114805	99,60
100	[20,30)	2,449655	2,312384	2,629054	99,58
100	[30,40)	8,650942	3,847853	21,112034	99,62
100	[40,50)	1,879981	0,225776	6,796908	99,65

Tabela 1: Resultados do ADIGR para o grupo $grd10$-49 variando o intervalo de densidade para cada valor de n.

Densidade(%)	Ordem	Média(s)	Mínimo(s)	Máximo(s)
[10,20)	25	0,002921	0,002406	0,005964
[10,20)	50	0,013724	0,012572	0,016522
[10,20)	75	0,045597	0,042210	0,045597
[10,20)	100	0,111050	0,109281	0,114805
[20,30)	25	0,002799	0,002426	0,005080
[20,30)	50	0,014696	0,012690	0,014696
[20,30)	75	0,048721	0,042091	0,072284
[20,30)	100	2,449655	2,312384	2,629054
[30,40)	25	0,002926	0,002435	0,008332
[30,40)	50	0,013920	0,012808	0,024368
[30,40)	75	0,573908	0,042770	0,907725
[30,40)	100	8,650942	3,847853	21,112034
[40,50)	25	0,002921	0,002458	0,008152
[40,50)	50	0,027170	0,012886	0,060788
[40,50)	75	0,197122	0,068596	0,496697
[40,50)	100	1,879981	0,225776	6,796908

Tabela 2: Resultados do ADIGR para o grupo $grd10$-49 variando n para cada intervalo de densidade.

gargalo no desempenho do AEPIG, algoritmo por eles proposto. Como no ADIGR a função *ssyevr_* é muito requisitada, quase todo o tempo de execução é dedicado a chamadas a essa função, como pode ser visto nas colunas **ssyevr(%)** das Tabelas 1 e 3. Além disso, o fato do autovetor principal possuir valores aproximados pode levar o ADIGR a podar um ramo da árvore que levaria a uma solução viável do problema, comportamento verificado em testes preliminares com grafos pequenos. Em contrapartida, o uso de uma função de precisão simples pode levar o algoritmo a descer por caminhos inviáveis. A solução neste caso seria o uso da função *dsyevr_*, levando efetivamente o algoritmo a um maior gasto de memória. Outra desvantagem em usar a função *ssyevr_* ou *dsyevr_*

Ordem	Média(s)	Mínimo(s)	Máximo(s)	ssyevr(%)
100	0,107396	0,106890	0,113412	99,82
150	0,458421	0,456938	0,467879	99,91
200	1,316710	1,314676	1,330748	99,92
250	3,120374	3,031284	3,769767	99,92
300	6,295035	6,183850	6,937087	99,95
350	20,825506	12,160134	30,234422	99,96
400	45,842670	19,649873	61,919300	99,98
450	72,474809	29,415859	114,539623	99,98

Tabela 3: Resultados do ADIGR para o grupo $grd04$.

é a necessidade de alocar uma matriz de ponto flutuante a cada chamada, uma vez que essas rotinas não preservam a matriz de adjacência passada como parâmetro.

É importante ressaltar que o ADIGR foi bem sucedido em testes realizados com as cinco instâncias isomorfas presentes em Dharwadker & Tevet (2009).

6. Conclusão e Propostas de Trabalhos Futuros

Neste trabalho foi investigada uma forma de utilização do autovetor principal de grafos na resolução do Problema de Isomorfismo de Grafos Regulares. Nesse sentido foi proposto um algoritmo de busca em profundidade na árvore de soluções, com poda, denominado ADIGR. Este algoritmo realiza n modificações equivalentes nos grafos de entrada e faz uso do autovetor principal para identificar assimetrias, podar a árvore de busca e restringir as possibilidades de mapeamento entre os vértices.

Os testes realizados mostram que o algoritmo proposto neste trabalho é consistente para a resolução do PIGR com instâncias compostas por grafos regulares não isomorfos e um pequeno conjunto de pares de grafos isomorfos. O desempenho do algoritmo não foi considerado satisfatório devido fundamentalmente ao tempo de processamento gasto com as chamadas da função de cálculo do autovetor principal, mostrando ser necessário investigar outras formas de obtenção desse autovetor.

Neste trabalho não é possível afirmar que o ADIGR é um algoritmo exato, pois não é provado matematicamente que os grafos G_1^n e G_2^n, resultantes da aplicação de modificações equivalentes em G_1 e G_2, respectivamente, preservam as mesmas codições de isomorfismo.

Para trabalhos futuros, além de analisar o comportamento do ADIGR com um maior grupo de instâncias compostas por grafos isomorfos de maior ordem e provar que trata-se de um algoritmo exato, serão investigadas outras formas de modificação de grafos que sejam menos custosas em termos de espaço e outras formas de obtenção do autovetor principal que sejam mais eficientes em termos de tempo, espaço e precisão, pois o ADIGR é extremamente dependente das autocentralidades. Além disso, será realizado um estudo de um procedimento de resolução do PIGR que utilize uma versão paramétrica do ADIGR, ou seja, uma versão que tenha como parâmetro um valor p, $0 < p < n$, com o objetivo de reduzir a quantidade de modificações equivalentes a serem realizadas nos grafos de entrada, o que seria uma forma de viabilizar o uso da abordagem proposta com grafos de maior ordem.

Referências

Abdulrahim, M. A. & Misra, M. (1998), 'A graph isomorphism algorithm for object recognition', *Pattern Analysis and Applications*, **1**, 189–201.

Abreu, N. M. M., Del-Vecchio, R. R., Vinagre, C. T. M. & Stevanović, D. (2007), *in* 'Introdução à Teoria Espectral de Grafos com Aplicações', Notas em Matemática Aplicada, Sociedade Brasileira de Matemática Aplicada e Computacional.

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', IEEE Computer Society Press, pp. 1172–1177.

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

Corniel, D. G. & Gotlieb, C. C. (1970), 'An efficient algorithm for graph isomorphism', *J. ACM*, **17**, 51–64.

Dharwadker, A. & Tevet, J. (2009), The graph isomorphism algorithm, *in* 'Proceedings of the Structure Semiotics Research Group'.

Diestel, R. (2006), *Graph Theory (3º Ed.)*, Springer.

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

Karp, R. M. (1972), Reducibility among combinatorial problems, *in* 'Complexity of computer computations', Plenum Press, New York, USA, pp. 85–103.

Luchi, D. & Rangel, M. C. (2010), Geração aleatória de grafos regulares, Notas de aulas de estudos dirigidos I: Problemas de otimização em redes ópticas, Programa de Pós-Graduação em Informática, Vitória, ES, Brasil.

Luks, E. M. (1982), 'Isomorphism of graphs of bounded valence can be tested in polynomial time', *Journal of Computer and System Science*, **25**(1), 42–65.

McKay, B. D. (1981), Practical graph isomorphism, *in* 'Congressus Numerantium', Vol. 30, pp. 45–87.

Miyazaki, T. (1996), The complexity of mckay's canonical labeling algorithm, *in* 'DIMACS Series in Discrete Mathematics and Theoretical Computer Science'.

Nandi, R. C. (2006), Isomorfismo de grafos aplicado à comparação de impressões digitais, Programa de Pós-Graduação em Informática, Universidade Federal do Paraná, Curitiba, PR, Brasil.

Oliveira, M. O. & Greve, F. G. P. (2005), 'Um novo algoritmo de refinamento para testes de isomorfismo em grafos', *Revista Eletrônica de Iniciação Científica (REIC)*, **5**(3), 140–148.

Porumbel, D. C. (2009), 'A polynomial graph extension procedure for improving graph isomorphism algorithms', *Computing Research Repository*, **abs/0903.0136**.

Presa, J. L. L. & Anta, A. F. (2009), Fast algorithm for graph isomorphism testing, *in* 'Proceedings of the 8th International Symposium on Experimental Algorithms', SEA '09, Springer-Verlag, Berlin, Heidelberg, pp. 221–232.

Rajasekaran, S. & Kundeti, V. (2009), 'Spectrum based techniques for graph isomorphism', *International Journal of Foundations of Computer Science*, **20**(3), 479–499.

Santos, P. L. F. (2010), Teoria espectral de grafos aplicada ao problema de isomorfismo de grafos, Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, Vitória, ES, Brasil.

Santos, P. L. F., Rangel, M. C. & Boeres, M. C. S. (2010), Teoria espectral de grafos aplicada ao problema de isomorfismo de grafos, *in* 'XLII Simpósio Brasileiro de Pesquisa Operacional', Vol. 1, pp. 1–12.

Sorlin, S. & Solnon, C. (2008), 'A parametric filtering algorithm for the graph isomorphism problem', *Constraints*, **13**, 518–537.

Spielman, D. A. (2007), Spectral graph theory and its applications, *in* 'FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science', IEEE Computer Society, Washington, DC, USA, pp. 29–38.