AN ADAPTED POWER METHOD FOR EIGENVECTOR COMPUTING APPLIED TO A GRAPH ISOMORPHISM ALGORITHM

Marcos Daniel Valadão Baroni
Maria Claudia Silva Boeres
Maria Cristina Rangel
Universidade Federal do Espírito Santo - UFES
Programa de Pós-Graduação em Informática
Departamento de Informática
Av. Fernando Ferrari, 514 - Goiabeiras - 29075-910 - Vitória - ES - Brasil
marcosdaniel.baroni@gmail.com, {boeres, crangel}@inf.ufes.br

RESUMO

O problema de isomorfismo de grafos pode ser aplicado em vários problemas da vida real. Este trabalho é sobre um algoritmo que resolve o problema utilizando conceitos da teoria espectral de grafos. O problema em questão é considerado para que uma versão otimizada do método de potência seja proposta para o cálculo da centralidade de autovetor, bem como a sua aplicação no algoritmo espectral original.

PALAVRAS CHAVES. Problema de Isomorfismo de Grafos. Centralidade de Autovetor. Cálculo de Autovetor.

Teoria e Algoritmos em Grafos

ABSTRACT

The graph isomorphism problem can be applied to many real-life issues. This work is about an algorithm for solving the problem using some concepts of spectral graph theory. The concerned problem is considered so that an optimized power method for computing the eigenvector centrality is proposed, as well as its application in the original spectral algorithm.

KEYWORDS. Graph Isomorphism Problem. Eigenvector Centrality. Eigenvector Computing.

Graph Theory and Algorithms

1 Introduction

The Graph Isomorphism Problem (GIP) can be applied in many real-life problems, for example, those involving pattern recognition (Conte et al. (2004)) and identification of structural similarities in chemical compounds (Oliveira & Greve (2005); Fortin (1996)). In the latter, it is necessary to determine whether or not a molecule has the same structure of another before giving it an exclusive name. A possible way of doing it is representing the molecules as graphs where each vertex represents an atom and the edges represent its chemical bonds. By doing that we can say that the molecules have a similar structure if their graphs are isomorphic.

Formally two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ of same order and size are isomorphic if there is a bijection $f : V_1 \rightarrow V_2$ such that their structural adjacencies are preserved, i.e., $\forall a, b \in V_1, (a, b) \in E_1 \Leftrightarrow \big(f(a), f(b)\big) \in E_2$. The GIP is to determine if two graphs are isomorphic (Diestel (2006); Dalcumune (2008)). Although necessary, the conditions of same order and size are not sufficient to answer if two graphs are isomorphic.

The GIP is one of the few problems that belongs to the class NP, but it is not known whether it is in class P or NP-complete, though it is not a co-NP problem (Fortin (1996); Jenner et al. (2003)). The commonly accepted assumption is that it is strictly between the two classes (Arvind & Torán (2005)).

There are several exact algorithms in the literature to solve the GIP we can highlight for their efficiency: the Ullmann algorithm (Ullmann (1976)), VF2 (Cordella et al. (2001)), Nauty (McKay (1981)) and Bliss (Junttila & Kaski (2007)), among others. All of them improve the search for the GIP solution using different filters.

There are also examples of polynomial time algorithms dedicated to specific classes of graphs (Sorlin & Solnon (2004); Uehara et al. (2005); Zager (2005); Dharwadker & Tevet (2009)). Moreover, we can cite Xiutang & Kai (2008) as an heuristic algorithm example using the Simulated Annealing to solve the GIP.

Spectral Graph Theory (SGT) is a field of Discrete Mathematics that treats graph properties using their matrix representation (adjacency, Laplacian, signless Laplacian, among others) and spectrum (Hogben (2009)). Concepts such as the spectrum of a graph and the eigenvector centrality have been used in graph theory for the identification of relevant information related to graphs. The SGT has aroused interest in many research groups on the last three decades due its wide application in several areas, such as chemistry, computer engineering and computer science (Abreu (2005)).

We are motivated with the study of filters for GIP algorithms. Lee (2007) use as a filter the usual one which is the vertices degree. Santos et al. (2010) present two others different filters using SGT concepts, proposing two theoretical results about these filters to isomorphism detection, which are included in a tree-search GIP procedure, originating the algorithm called SAGIP. However, according to the results presented in that paper, the eigenvector computing required to the use of the proposed filters, using a available package of linear algebra functions, increases significantly the algorithm CPU time.

In this work, we propose an adaptation of a well known method from the literature for computing eigenvectors, called power method (Saad (1992)), in order to improve the CPU time spent by the SAGIP algorithm proposed in Santos et al. (2010).

The following section introduces some concepts about spectral graph theory. Section 3 explains the original algorithm (SAGIP). In Section 4 an adaptation to the power method, as well as the Adapted SAGIP are proposed. Section 5 contains some computational results and in Section 6 we present the conclusions about this work.

2 Some basic concepts of Spectral Graph Theory

In this section we present several definitions about SGT that will support some issues addressed in this work. The definitions described here were extracted from Abreu et al. (2007), in which can also be found the related mathematical proofs.

Let $G = (V, E)$ be an undirected simple graph, with n vertices and m edges. The $A_{n\times n}$ matrix where the entry a_{uv} is 1 if $(u, v) \in E$ and is 0 otherwise is defined as **adjacency matrix** of G.

For a graph G with adjacency matrix $A(G)$, the **characteristic polynomial** of G is defined as $p_G(\lambda) = det\big(A(G) - \lambda I\big)$, where the root λ of the polynomial is called **eigenvalue** of G. As G has n vertices it also has n eigenvalues, the largest being called **index** of the graph. Also, given a nonzero vector v such as $A(G)v = \lambda v$ is an **eigenvector** associated to λ. The eigenvector associated to the index is called **dominant eigenvector**.

The **spectrum** of G, denoted by $spect(G)$, is defined as a $2 \times d$ matrix, having in its first row the d distinct eigenvalues arranged in descending order and in its second row their respective algebraic multiplicities. Consequently, the first spectrum entry corresponds to G index, usually denoted by λ_1.

Two graphs G_1 and G_2 are **cospectral graphs** if their eigenvalues are the same, i.e., $spect(G_1) = spect(G_2)$. From this definition we conclude that if two graphs are isomorphic they are cospectral, however the converse is not always true.

Considering x the eigenvector associated to graph index, the component x_i is the **eigenvector centrality** of the i^{th} vertex.

3 A Spectral Based Exact Algorithm for GIP

One of the Santos et al. (2010) motivations is the use of the centrality concept as an additional filter (as it is a graph invariant) in the detection of isomorphic graphs, since results from the literature state that two vertices of same degree in a graph may have distinct centrality values (Grassi et al. (2007)). Thus, two theoretical results based on the preservation of centrality in isomorphic graphs have been proposed and proved in Santos (2010). The first concerns graphs associated to equal eigenvector centralities with all components distinct from each other are isomorphic. The second says that isomorphic graphs have proportional eigenvector centralities. Both theorems are used as filters in their algorithm, which contains three phases. In the first phase the graphs eigenvector centralities are computed. The purpose of computing these eigenvectors is to gather the vertices in groups of centrality, thus the associations are restricted to vertices of the same group. Until in this stage, the second result cited above is checked. In the second phase, the first result is verified and in the last, the centrality groups produced in phase 1 are used as a filter to the solution tree built according to the exact *backtracking* algorithm presented in Lee (2007). When the tree search ends in a leaf, a feasible association is found and the algorithm halts concluding the graphs are isomorphic. Otherwise, the solution tree search is exhausted ending in the tree root, no feasible association is found and the algorithms halts concluding the graphs are not isomorphic. The algorithm called SAGIP (Spectral Algorithm for the Graph Isomorphism Problem) is reproduced in Algorithm 1.

4 Computing The Eigenvector Centrality

As seen in previous section the proposed algorithm builds vertex association groups using their eigenvector centralities. So a crucial issue is finding an efficient method to do that, i.e., to calculate the dominant eigenvector of a graph adjacency matrix. The method proposed here is an adapted version of the power method (Saad (1992)) for graphs adjacency matrices.

Algorithm 1: **The Spectral Algorithm for the Graph Isomorphism Problem (SAGIP)**

Input: The adjacency matrices $A(G_1)$ and $A(G_2)$ from graphs G_1 and G_2
Output: true (if $G_1 \simeq G_2$) or false (otherwise)

```
1  begin
2      Compute the centrality eigenvectors x¹, x² of G1, G2 ;        // Phase 1
3      Order x¹ and x² components in ascending order
4      if x¹ ≠ k x², k ∈ R* then
5          return false
6      else
7          if x^i = (x_1^i, ..., x_n^i), such as x_j^i ≠ x_k^i,
8             j, k = 1, ..., n, j ≠ k and i = 1, 2 ;                 // Phase 2
9          then
10             return true
11         else
12             Execute backtracking using vertices grouped by centrality ;   // Phase 3
13             if found a feasible solution
14             then
15                 return true
16             else
17                 return false
18 end
```

4.1 The Power Method

The power method is probably the simplest known iterative eigenvector computing method. It computes a sequence of k $A^k v_o$ vectors where v_0 is a nonzero initial vector and A is a $n \times n$ matrix. Those vectors converge[1] to the eigenvector associated with the largest modulus eigenvalue [2] (Saad (1992)). The power method algorithm is presented in Algorithm 2.

Algorithm 2: **The Power Method.**

Input: A: $n \times n$ matrix, $\mathbf{v_0}$ initial solution
Output: λ_1: the eigenvalue, $\mathbf{v}^{\lambda_1}$: the eigenvector

```
1  begin
2      for i ← 1 to k do
3          w_i = A v_{i-1} ;          // product:  n² floating operations
4          α_i = max(w_i) ;           // find max:  n floating comparisons
5          v_i = w_i / α_i ;          // normalization:  n floating operations
6      λ1 = α_k ;                                          // the eigenvalue
7      v^{λ1} = v_k ;                                      // the eigenvector
8  end
```

The normalization factor α_i is the component of Av_{i-1} of which has the maximum modulus. The algorithm does not converge if the dominant eigenvalue is complex and both the original matrix and the initial vector are real. At the k^{th} iteration the v_k vector is generated as:

[1] given $\epsilon > 0$, $|v_k - v_{k-1}| < \epsilon$
[2] which is the graph index λ_1 if taken its adjacency matrix.

$$v_k = \frac{1}{\alpha_k} A v_{k-1} = \frac{1}{\alpha_k}\frac{1}{\alpha_{k-1}} A^2 v_{k-2} = \underbrace{\frac{1}{\alpha_1 \alpha_2 \cdots \alpha_{k-1}\alpha_k}}_{\text{normalization factor}} A^k v_0 \qquad (1)$$

The Figure 1 exemplifies the computed sequence of vectors.

$$\underset{A}{\begin{bmatrix} 0&1&1&0&0&0\\ 1&0&1&0&1&0\\ 1&1&0&1&0&0\\ 0&0&1&0&0&0\\ 0&1&0&0&0&1\\ 0&0&0&0&1&0 \end{bmatrix}}, \underset{v_0}{\begin{bmatrix} 1.0\\ 1.0\\ 1.0\\ 1.0\\ 1.0\\ 1.0 \end{bmatrix}} \underset{v_1}{\begin{bmatrix} 0.667\\ 1.000\\ 1.000\\ 0.333\\ 0.667\\ 0.333 \end{bmatrix}} \underset{v_2}{\begin{bmatrix} 0.857\\ \mathbf{1.000}\\ 0.857\\ \mathbf{0.429}\\ \mathbf{0.571}\\ \mathbf{0.286} \end{bmatrix}} \underset{v_3}{\begin{bmatrix} \mathbf{0.813}\\ 1.000\\ 1.000\\ \mathbf{0.375}\\ \mathbf{0.563}\\ \mathbf{0.259} \end{bmatrix}} \underset{v_4}{\begin{bmatrix} \mathbf{0.842}\\ \mathbf{1.000}\\ \mathbf{0.921}\\ \mathbf{0.421}\\ \mathbf{0.526}\\ \mathbf{0.237} \end{bmatrix}} \cdots \underset{v_k}{\begin{bmatrix} \mathbf{0.843}\\ \mathbf{1.000}\\ \mathbf{0.967}\\ \mathbf{0.414}\\ \mathbf{0.525}\\ \mathbf{0.225} \end{bmatrix}}$$

Figure 1: An example of some of the power method iterations.

The normalization step is important because (a) it prevents the unlimited growth of the v_i components – reducing its maximum modulus component to 1 – and (b) after k iterations α_i converges to the λ_1 eigenvalue (Saad (1992)).

4.2 The Adapted Power Method

According to the conclusions presented in Santos et al. (2010), the authors observed that one of the principle bottlenecks of its algorithm CPU time is the eigenvector computing required to the use of filters. In that work, this computing is performed by functions of CLAPACK, a C programming language version of the LAPACK (Linear Algebra PACKage) a library of Fortran 77 subroutines for solving most commonly occurring problems in numerical linear algebra (Anderson et al. (1999)).

However, for the purpose of detecting graph isomorphism, we need only the eigenvector associated to the graph index λ_1, while the set of CLAPACK functions are prepared to calculate in addition, several other informations related to the matrix spectrum.

Thus, some peculiarities of the graph adjacency matrix and the GIP will now be considered, hence some optimizations will be formulated for the power method previously presented.

4.2.1 Reducing The Number of Iterations

In section 3 we exposed that the objective of taking the eigenvector centrality is to determine groups of vertices associated with each distinct value of eigenvector components, building the groups of vertices candidates to map with the other graph.

The greater the number of iterations of the power method, the more accurate is the eigenvector calculated. A first attempt to reduce the CPU time of the power method is to decrease its number of iterations because it has been empirically observed that the suitable number of iterations to build a robust filter over non-regular graph is smaller than the default, specially for large/dense graphs. For building this filter the eigenvector component values are not relevant, the important issue is if they are different or not from each other. It allow us to interrupt the power method process of convergence earlier as long as the same number of iterations are used for both graphs.

Moreover it is not difficult to conclude that the components of the vector v_1 (resulting from the the first iteration with $v_0 = \mathbf{1}$) correspond to the graph vertices degree. If they are already known we can use them as the initial vector, saving an iteration (Figure 2).

4.2.2 Skipping Normalization Step

Another optimization considered for the power method in the GIP context is to ignore the v_i vector normalization step. This would save us n floating point comparisons and n floating point operations on every iteration (see line 5 and 6 respectively of Algorithm 2). It would even allow us to use just integer values for the v_i vectors and upon this, easier arithmetic operations.

The first consequence of this optimization is letting the v_i vectors components grow unlimited. As seen in section 4.2.1 the number of iterations has been significantly reduced for the proposed method. Therefore the growth of the vectors components is not a problem, at least for some quite iterations[3]. The other consequence is not computing the eigenvalue which is not a problem either because the proposed GIP algorithm does not really need it. Hence we can substitute the original method floating point vectors by integer vectors, since they still represent the eigenvector without the normalization factor (Eq. 1). The Figure 2 shows an example of some iterations without the normalization step.

$$\underset{A}{\begin{bmatrix} 0&1&1&0&0&0\\ 1&0&1&0&1&0\\ 1&1&0&1&0&0\\ 0&0&1&0&0&0\\ 0&1&0&0&0&1\\ 0&0&0&0&1&0 \end{bmatrix}}, \quad \underset{v_0}{\begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix}} \quad \underset{v_1}{\begin{bmatrix}2\\3\\3\\1\\2\\1\end{bmatrix}} \quad \underset{v_2}{\begin{bmatrix}6\\\mathbf{7}\\6\\\mathbf{3}\\\mathbf{4}\\\mathbf{2}\end{bmatrix}} \quad \underset{v_3}{\begin{bmatrix}\mathbf{13}\\16\\16\\\mathbf{6}\\\mathbf{9}\\\mathbf{4}\end{bmatrix}} \quad \underset{v_4}{\begin{bmatrix}\mathbf{32}\\\mathbf{38}\\\mathbf{35}\\\mathbf{16}\\\mathbf{20}\\\mathbf{9}\end{bmatrix}}$$

Figure 2: An example of some of the modified power method iterations.

4.2.3 Optimized Matrix-vector Product

The adjacency matrices of the graphs related to GIP are binary by definition. It is possible to consider these matrices always being sparse because if they are not (representing high density graphs) the GIP may be applied on the complements of both graphs instead. Thus, the matrix-vector product can be optimized by using the graph adjacency list representation in place of the matrix itself. Doing it, the matrix-vector product can be computed with m operations instead of n^2 (line 4 of Algorithm 2). Considering a $G = (V, E)$, for each vertex $v_i \in V$, the adjacency list structure is defined as a list of lists containing the vertices adjacent to v_i. Figure 3 exemplifies the technique. We can observe that for each line from the adjacency matrix, the product operations are performed only over its nonzero coefficients when considering the adjacency list representation.

$$\begin{bmatrix} 0&1&1&0&0&0\\ 1&0&1&0&1&0\\ 1&1&0&1&0&0\\ 0&0&1&0&0&0\\ 0&1&0&0&0&1\\ 0&0&0&0&1&0 \end{bmatrix} \cdot \begin{bmatrix}v_1\\v_2\\v_3\\v_4\\v_5\\v_6\end{bmatrix} = \begin{bmatrix}v_2+v_3\\v_1+v_3+v_5\\v_1+v_2+v_4\\v_3\\v_2+v_6\\v_5\end{bmatrix}$$

(a) using standart matrix representation

$$\begin{pmatrix}\{2,3\}\\\{1,3,5\}\\\{1,2,4\}\\\{3\}\\\{2,6\}\\\{5\}\end{pmatrix} \odot \begin{bmatrix}v_1\\v_2\\v_3\\v_4\\v_5\\v_6\end{bmatrix} \Rightarrow \begin{bmatrix}v_2+v_3\\v_1+v_3+v_5\\v_1+v_2+v_4\\v_3\\v_2+v_6\\v_5\end{bmatrix}$$

(b) using adjacency list representation

Figure 3: An example of a matrix-vector product.

Considering the modifications for Algorithm 2 proposed in this section, the adapted power method is presented in Algorithm 3. The $\odot$ operator means the product of the adjacency list by

[3] specially for modern architectures with 32 and 64 bits integers.

the vector and the functions $degrees(G)$ and $countGroups(v)$ give, respectively, the degree vector associated to G and the number of different components of vector v.

Algorithm 3: **The Adapted Power Method**

Input: $G = (V, E)$: graph (adjacency list structure)
Output: v: the non-converged centrality eigenvector

```
1  begin
2      v_0 = degrees(G);
3      ngroups_old ← countGroups(v_0);
4      v_1 = G_1 ⊙ v;
5      ngroups_new ← countGroups(v_1);
6      i ← 2;
7      while ngroups_old < ngroups_new and ngroups_new < |V| do
8          ngroups_old ← ngroups_new;
9          v_i = G_1 ⊙ v;
10         ngroups_new ← countGroups(v_i);
11         i ← i + 1;
12     v = v_i;
13 end
```

The algorithm initializes with the degree vector of G and its number of different components (lines 2 and 3). Then this vector is iteratively modified until the number of different components is unchanged or equal to $|V|$ (lines 4 to 12).

In order to improve the SAGIP (Algorithm 1) CPU time its line 2 was substituted by Algorithm 3 originating the Adapted Spectral Algorithm for the Graph Isomorphism Problem (ASAGIP).

5 Computational Results

For the purpose of comparing ASAGIP performance with some well known algorithms of the literature for GIP resolution, we also executed computational tests for the algorithms SAGIP, Bliss, Nauty and VF2.

All tests were performed on the same graph instances set used in Santos (2010), extracted from the Graph Database CD available in SIVALab (2001). The graph instances set is composed by 3000 couples of randomly isomorphic connected graphs, divided into three edge density groups $\eta = 0.01$, $\eta = 0.05$ and $\eta = 0.1$, respectively denoted by $r001$, $r005$ and $r01$. Each group contains 100 pairs of graphs (instances) of sizes 20, 40, 60, 80, 100, 200, 400, 600, 800 and 1000 vertices, amounting 1000 pairs of graphs in each group. According to the meaning of η, if n is the total number of nodes of the graph, the number of its edges will be equal to $\eta\left[\frac{n(n-1)}{2}\right]$. However, if this number is not sufficient to obtain a connected graph (i.e. at least n-1 edges), further edges are suitably added until the graph being generated becomes connected. Besides these instances, a new edge density group with $\eta = 0.5$ was generated using the Nauty random graph generator tool called *genrang*. We denoted this new group by $d05$.

The proposed algorithm was implemented in C Programming Language. The SAGIP used the function `dsyevr_` from CLAPACK 3.2.1 to calculate the eigenvector centrality. For the experimental tests Bliss and VF2 algorithms were extracted from iGraph 0.5.1 Library and Nauty 2.4 was used. The function `gettimeofday` from `time.h` C library measured all the algorithms execution time. The tests were performed on a machine with Intel® Core™2 Duo E4500 2.20GHz (2MB cache) processor and 2GB of RAM using Linux Ubuntu 10.04 OS kernel 2.6.32-30.

The graphics presented in Figures 4, 5, 6 and 7 show the CPU time average over the 100 instances of each graph size in each density group $r001$, $r005$, $r01$ and $d05$ respectively, for the five algorithms compared.

Observing the graphs from Figures 4, 5, 6 and 7, we can conclude that the optimizations proposed here improved significantly the efficiency of the power method and, in all tests performed, the ASAGIP algorithm obtained better results than SAGIP (in average it is 99% faster). In respect to all algorithms compared, the ASAGIP also obtained better results except by the Nauty Algorithm on the graph instances group $r001$. But even in this case the CPU times achieved were very closed.

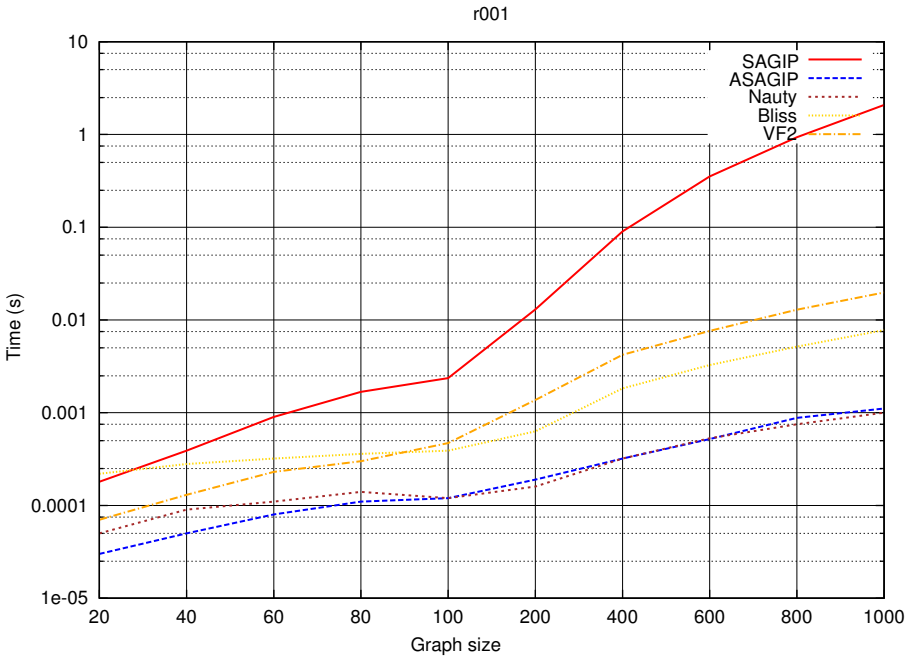

Figure 4: Result graphic for r001 class.

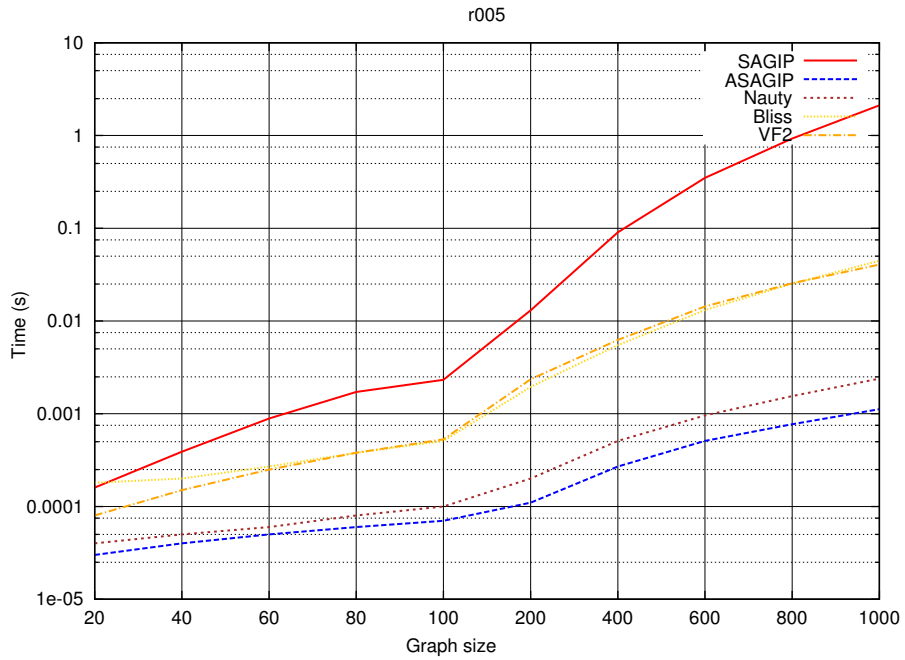

Figure 5: Result graphic for r005 class.

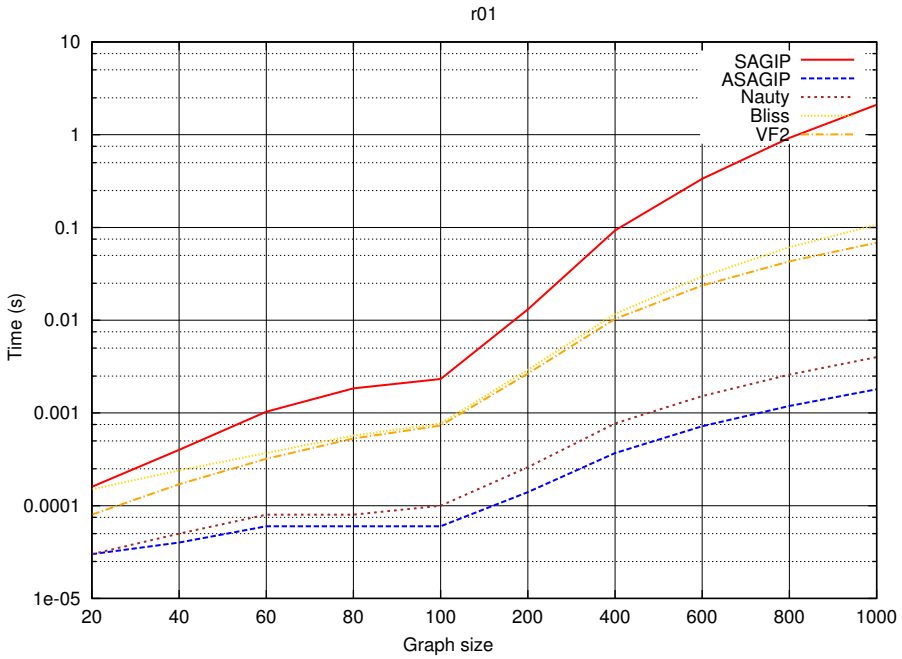

Figure 6: Result graphic for r01 class.

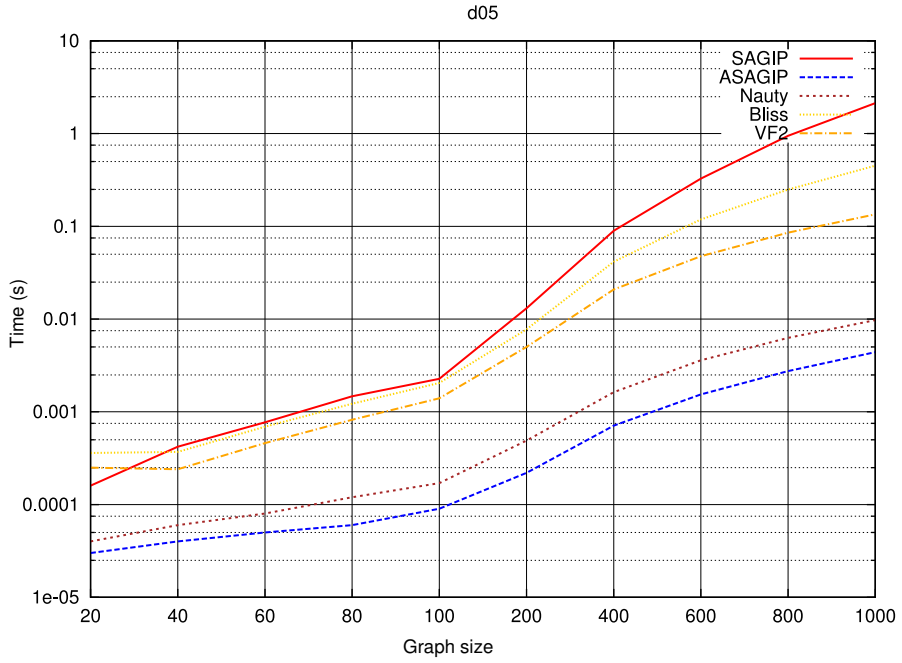

Figure 7: Result graphic for d05 class.

6 Concluding Remarks and Future Work

In this work we proposed an adapted power method for computing the graph eigenvector centralities as a filter to solve the graph isomorphism problem. It was motivated by the bottleneck for computing the eigenvector observed on the Spectral Algorithm for GIP (SAGIP) proposed in Santos et al. (2010). Considering the problem in question several optimizations were proposed for the power method originating the Adapted Spectral Algorithm for GIP (ASAGIP).

We concluded in this work that the proposed adaptations for the power method are very efficient, especially when the graphs becomes larger and denser in which less iterations is needed to classify the vertices in different groups of centrality.

Another interesting conclusion is about the number of iterations needed for the adapted power method. For low density graphs ($r001$) this number was relatively high (average of 4.8 iterations) needing 7 iterations for some of the instances. However we observed that this number was much smaller for denser graphs, e.g. for group $r01$ the average iteration number was 3.1, needing a maximum of 4 iterations for some of its instances.

As future work we intend to investigate the relation between the eigenvector centralities and the density of the graphs. It seems that vertices in larger and denser graphs tend to have more distinct eigenvector centralities. We also intend to study how the method could be applied to regular graphs as well as optimized for sparse graphs.

References

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*. Sociedade Brasileira de Matemática Aplicada e Computacional.

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling, S., McKenney, A., , & Sorensen, D. (1999). Lapack user's guide. `http://www.netlib.org/lapack/lug/`.

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

Conte, D., Foggia, P., Sansone, C., & Vento, M. (2004). Thirty years of graph matching in pattern recognition. *IJPRAI*, 18(3):265–298.

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.

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

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

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

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.

Junttila, T. & Kaski, P. (2007). Engineering an efficient canonical labeling tool for large and sparse graphs. In Applegate, D., Brodat, G. S., Panario, D., & Sedgewick, R., editors, *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments and the Fourth Workshop on Analytic Algorithms and Combinatorics*. SIAM.

Lee, L. (2007). Reformulação do problema de isomorfismo de grafos como um problema quadrático de alocação. Master Thesis, Programa de Pós-Graduação em Informática - UFES, Vitória, ES.

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. *XXV Congresso da Sociedade Brasileira de Computação*.

Saad, Y. (1992). *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press, Manchester, UK.

Santos, P. L. F. (2010). Teoria espectral de grafos aplicada ao problema de isomorfismo de grafos. Master Thesis, Programa de Pós-Graduação em Informática, UFES, Vitória, ES.

Santos, P. L. F., Rangel, M. C., & Boeres, M. C. S. (2010). Teoria espectral de grafos aplicada ao problema de isomorfismo de grafos. In ILTC, editor, *Proceedings of XLII Simpósio Brasileiro de Pesquisa Operacional (SBPO 2010)*, pages 1–12, Bento Gonçalves, RS.

SIVALab (2001). The Graph Database CD. `http://amalfi.dis.unina.it/graph/doc/graphdb.html`. A huge collection of graphs realized by the *Intelligent Systems and Artificial Vision Lab. (SIVALab) of the University of Naples "Federico II"*.

Sorlin, S. & Solnon, C. (2004). A global constraint for graph isomorphism problems. In Springer-Verlag, editor, *the 6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems (CP-AI-OR 2004)*, pages 287–301.

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. Master Thesis, Department of Electrical Engineering and Computer Science. Massachusetts Institute of Technology.