IEICE TRANS. INF. & SYST., VOL.E95–D, NO.11 NOVEMBER 2012

LETTER

Detecting Non-subgraphs Efficiently by Comparing Eigenvalues of Decomposed Graphs

Kaoru KATAYAMA[†a)], ***Member***, **Yosuke AMAGASA**[††], ***and*** **Hideki NAGAYA**[††], ***Nonmembers***

SUMMARY The problem of deciding whether a graph contains another graph appears in various applications. For solving this problem efficiently, we developed a numerical method to detect non-subgraphs, graphs which are not subgraphs of other graphs, by comparing eigenvalues of graphs. In this paper, we propose a method to make the detection more efficient by comparing of eigenvalues of graphs decomposed according to labels of the vertices and the edges. The new approach not only reduces the cost of computing eigenvalues but also increases the possibility of detecting non-subgraphs. The experimental evaluation shows the effectiveness of the proposed method.

key words: *subgraph, interlace, eigenvalue*

1. Introduction

Graphs are used as a data model in various applications such as chemical compounds, logic circuits and social networks. The problem of deciding whether a graph contains another graph appears commonly in such applications. This is called the subgraph isomorphism problem and known to be NP-complete. If graphs are large, it is hard to solve this problem with combinatorial methods. We proposed a numerical method for checking whether a graph g^s is *not* a subgraph of another graph g in Nagaya et al. [1]. We call such a graph g^s a non-subgraph. Although it is not possible to confirm that g^s is a subgraph of g by the method, we can potentially reduce the work of solving the problem with time-consuming combinatorial methods by using it as preprocessing. The method is based on the interlace theorem [2] on eigenvalues of a symmetric matrix and its submatrix.

In this paper, we propose a method to make detect non-subgraphs more efficient by comparing eigenvalues of graphs which are decomposed according to labels of the vertices and the edges, instead of comparing the eigenvalues of the original graphs. This approach not only reduces the cost of computing eigenvalues but also increases the chance of finding non-subgraphs. It is also suitable for parallel processing. We compare the processing time and the performance of detecting non-subgraphs of the proposed method with our previous method and VF2 [3] by experiment. VF2 is one of the state-of-the-art algorithms for finding subgraphs. The result shows the effectiveness of our method clearly.

Manuscript received March 30, 2012.
Manuscript revised July 21, 2012.

[†]The author is with the Graduate School of System Design, Tokyo Metropolitan University, Hino-shi, 191–0065 Japan.
[††]This work was partially done while the authors were with the Graduate School of System Design, Tokyo Metropolitan University, Hino-shi, 191–0065 Japan.
a) E-mail: kaoru@tmu.ac.jp

DOI: 10.1587/transinf.E95.D.2724

2. Related Work

Many algorithms have been proposed to solve the subgraph isomorphism problem. Ullmann [4] presented an algorithm based on backtracking technique which reduces the search space efficiently. VF2 [3] proposed by Codella et al. is also a backtracking algorithm. McKay [5] proposed an algorithm for solving the graph isomorphism problem. Eigenvalues of graphs are used for indexing graphs. Shokoufandeh et al. [6] proposed an indexing method for object recognition. It maps the tree representing features of an object into a vector space by eigenvalues of the adjacency matrix of the tree. Zhang et al. [7] and Zou et al. [8] proposed indexing methods using eigenvalues of graphs for XML documents and for graphs, respectively. Zhang et al. represented an XML document and an XPath query as graphs and used the maximum eigenvalues and the minimum ones of the graphs as their features. They use interlacing property between the maximum eigenvalue and the minimum one to process a query. Zou et al. also used some eigenvalues of a graph as its feature. Their query processing is based on the relation between the maximum and the second largest eigenvalues of a graph in a database and a query graph. For classifying large graphs according to large common induced subgraphs as a similarity measure, Vinh et al. [9] propose a graph kernel based on eigenvalues of graphs and the interlace theorem. They also propose an algorithm for optimizing the eigenvalues to obtain better classification accuracy.

3. Preliminaries

3.1 Graphs and Matrices

A *directed labeled graph* g is a tuple (V, E, L, μ) where V is a set of vertices, E is a set of edges, L is a set of labels of vertices and edges, and μ is a labeling function $V \cup E \to L$. An edge $e \in E$ is an ordered pair (v_1, v_2) of vertices in V. We also denote a set of vertices V of g as $V(g)$ and a set of edges E of g as $E(g)$. If, for any $(v_1, v_2) \in E$, there is the edge $(v_2, v_1) \in E$, g is called an *undirected* labeled graph. For a vertex v of g, if there is not an edge (v, v') or $(v', v) \in E(g)$, we call v an *isolated vertex*. We assume that labels of vertices and edges are real numbers. If they are not real numbers in a practical application, we change each label to a real number

in some suitable way.

Definition 1: (Subgraph) A graph $g^s = (V^{g^s}, E^{g^s}, L^{g^s}, \mu^{g^s})$ is a *subgraph* of another graph $g = (V^g, E^g, L^g, \mu^g)$, if there is an injection $i : V^{g^s} \to V^g$ which satisfies the following conditions for any $(v_1, v_2) \in E^{g^s}$.

- $(i(v_1), i(v_2)) \in E^g$
- $\mu^{g^s}(v_1) = \mu^g(i(v_1))$ and $\mu^{g^s}(v_2) = \mu^g(i(v_2))$
- $\mu^{g^s}(v_1, v_2) = \mu^g(i(v_1), i(v_2))$

The injection i is called *subgraph isomorphism*.

Definition 2: (Induced Subgraph) Let a graph $g^s = (V^{g^s}, E^{g^s}, L^{g^s}, \mu^{g^s})$ be a subgraph of a graph $g = (V^g, E^g, L^g, \mu^g)$ where the subgraph isomorphism is $i : V^{g^s} \to V^g$. g^s is an *induced subgraph*, if it satisfies the following condition for any pair v_1 and v_2 of vertices in V^{g^s}.

- if there is an edge $(i(v_1), i(v_2))$ in E^g, there is the edge (v_1, v_2) in E^{g^s}.

Definition 3: An adjacency matrix $A^g = (a^g_{ij})$ of a graph $g = (V^g, E^g, L^g, \mu^g)$ is the $|V^g| \times |V^g|$ matrix as follows.

$$a^g_{ij} := \begin{cases} \mu^g(v_i) & \text{if } i = j \text{ for } v_i \in V^g, \\ \mu^g(v_i, v_j) & \text{if } i \neq j \text{ for } (v_i, v_j) \in E^g, \\ 0 & \text{otherwise.} \end{cases}$$

$\mu(v_i)$ and $\mu(v_i, v_j)$ are real numbers in this paper.

3.2 Interlace Theorem and Induced Subgraphs of Undirected Graphs

Haemers [2] states the interlace theorem as follows.

Definition 4: Let $\{\alpha_i\}_{i=1,\ldots,n}$ and $\{\beta_j\}_{j=1,\ldots,m}$ be two ordered sequences of real numbers where $m < n$, $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_n$ and $\beta_1 \leq \beta_2 \leq \ldots \leq \beta_m$, respectively. We say that $\{\beta_j\}_{j=1,\ldots,m}$ *interlaces* $\{\alpha_i\}_{i=1,\ldots,n}$, if the following condition is satisfied for $k = 1, \ldots, m$.

$$\alpha_k \leq \beta_k \leq \alpha_{k+(n-m)}$$

Theorem 1: (Interlace Theorem) Given a real $n \times m$ matrix S such that $S^T S = I$ and a symmetric $n \times n$ matrix A, the eigenvalues of a $m \times m$ matrix $S^T A S$ interlace those of A.

When a graph g^s is an induced subgraph of an undirected graph g, both of the adjacency matrices A^g and A^{g^s} are symmetric and there exists the matrix S such that $A^{g^s} = S^T A^g S$ and $S^T S = I$. This means that, if the eigenvalues of A^{g^s} *do not* interlace the eigenvalues of A^g, the graph g^s is *not* an induced subgraph of the undirected graph g.

4. Finding Non-subgraphs by Eigenvalues

Given two graphs g and g^s, we find whether g^s is not a subgraph of g according to the following steps in Nagaya et al. [1]. At first, we reduce the sizes of g and g^s by comparing labels of vertices and edges and deleting unnecessary edges and vertices. Then, we check whether eigenvalues of g^s interlace eigenvalues of g. In the following sections, g and g^s are undirected labeled graphs.

4.1 Matrix Representation for Subgraphs

In order to use the interlace theorem for finding graphs which are not subgraphs of a graph, we need to represent two graphs which we compare as symmetric matrices. Adjacency matrices of directed graphs are not generally symmetric. In addition, although adjacency matrices of undirected graphs are symmetric, for the adjacency matrices A^g and A^{g^s} of an undirected graph g and its subgraph g^s, there does not exist a matrix S which satisfies the assumption of the interlace theorem, that is, $A_{g^s} = S^T A_g S$ and $S^T S = I$, in general. So we use the following matrix representation of a graph. It is a modification of the matrix representation given by Haemers [2].

Definition 5: An *extended incidence matrix* M^g of a graph $g = (V^g, E^g, L^g, \mu^g)$ is a matrix

$$\begin{bmatrix} P & N \\ N^T & Q \end{bmatrix}$$

where $N = (n_{ij})$, $P = (p_{ij})$ and $Q = (q_{ij})$ are $|V^g| \times |E^g|$, $|V^g| \times |V^g|$ and $|E^g| \times |E^g|$ matrices as follows, respectively.

$$n_{ij} := \begin{cases} 1 & \text{if } (v_i, v_k) = e_j \in E^g \text{ for some } k, \\ 0 & \text{otherwise.} \end{cases}$$

$$p_{ij} := \begin{cases} \mu^g(v_i) & \text{if } i = j \text{ for } v_i \in V^g, \\ 0 & \text{otherwise.} \end{cases}$$

$$q_{ij} := \begin{cases} \mu^g(e_i) & \text{if } i = j \text{ for } e_i \in E^g, \\ 0 & \text{otherwise.} \end{cases}$$

N is an incidence matrix of g. N^T is the transpose of N.

4.2 Reducing Size of Graphs

When we check whether a graph g contains another graph g^s, it is clear that g's vertices and edges with the labels which are not contained in g^s are not necessary. By deleting such vertices and edges from g, we can reduce the size of g and the cost of computing the eigenvalues of g. Moreover we find that g^s is not a subgraph of g, if the number of g^s's edges or vertices with a label is more than the number of g's edges or vertices with the same label.

4.3 Comparing Eigenvalues of Graphs

We illustrate the algorithm based on the bisection method for comparing eigenvalues with an example. We denote the ordered sequence of eigenvalues of a graph g as $\{eig(g)_i\}_{i=1,\ldots,n}$ where $eig(g)_1 \leq eig(g)_2 \leq \ldots \leq eig(g)_n$. For the ith eigenvalue $eig(g)_i$, $eig(g)^{low}_i$ and $eig(g)^{up}_i$ are the lower and upper bounds, respectively. For all eigenvalues of g, we denote the lower and upper bounds as $low(eig(g))$ and $up(eig(g))$, respectively.

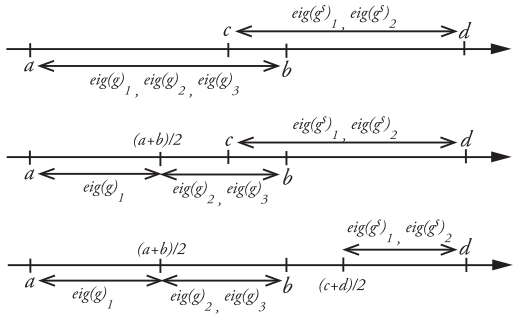

Fig. 1 Updating ranges of eigenvalues $\{eig(g)_i\}_{i=1,2,3}$ and $\{eig(g^s)_i\}_{i=1,2}$ of two graph g and g^s.

Example 1: Figure 1 shows how our algorithm updates the ranges of eigenvalues $\{eig(g)_i\}_{i=1,2,3}$ and $\{eig(g^s)_i\}_{i=1,2}$ of two graphs g and g^s. The first figure shows that the ranges of $\{eig(g)_i\}_{i=1,2,3}$ and $\{eig(g^s)_i\}_{i=1,2}$ overlap, and the lower and upper bounds, $low(eig(g))$, $up(eig(g))$, $low(eig(g^s))$ and $up(eig(g^s))$ are a, b, c and d, respectively. We try to compare the eigenvalues $eig(g)_1$ and $eig(g^s)_1$ at first. Since $eig(g)_1^{up} - eig(g)_1^{low} = b - a > eig(g^s)_1^{up} - eig(g^s)_1^{low} = d - c$ here, we narrow the range of $eig(g)_1$. The second figure shows that $eig(g)_1^{up} = (a + b)/2 < eig(g^s)_1^{low} = c$. Therefore $eig(g)_1$ is less than $eig(g^s)_1$ and the condition of the interlace theorem is satisfied for $eig(g)_1$ and $eig(g^s)_1$. Next we try to compare $eig(g)_2$ and $eig(g^s)_1$. Since the ranges of $eig(g)_2$ and $eig(g^s)_1$ overlap and $eig(g^s)_1^{up} - eig(g^s)_1^{low} = d - c > eig(g)_2^{up} - eig(g)_2^{low} = b - (a + b)/2$, we narrow the range of $eig(g^s)_1$. The third figure shows that $eig(g)_2^{up} < eig(g^s)_1^{low}$. Therefore $eig(g)_2$ is less than $eig(g^s)_1$ and the condition of the interlace theorem is not satisfied for $eig(g)_2$ and $eig(g^s)_1$. This means that g^s is not a subgraph of g.

5. Comparing Eigenvalues of Decomposed Graphs

Let a graph g^s be a subgraph of another graph g. Then, the graphs into which g^s is decomposed according to a label are also subgraphs of the graphs into which g is decomposed according to the same label. We propose a method to detect non-subgraphs more efficiently by comparing eigenvalues of graphs decomposed according to labels of the vertices and the edges, instead of comparing the eigenvalues of the original graphs, g and g^s. After the decomposition, we check whether eigenvalues of the decomposed graphs of g^s interlace eigenvalues of the decomposed graphs of g with the same labels. If eigenvalues of one of such decomposed graphs of g^s do not interlace the eigenvalues of the corresponding decomposed graph of g, we do not need to compute and compare eigenvalues of other pairs of decomposed graphs. This lowers the cost of computing the eigenvalues of the graphs and improves the likelihood of detecting whether g^s is not a subgraph of g. This approach is also suitable for parallel processing. Although there are various ways to decompose graphs according to labels, we consider the following ones. For a graph, the set of the decomposed graphs varies depending on the order of decomposition using labels of vertices and edges.

D1 decomposing graphs according to only labels of vertices
D2 decomposing graphs according to only labels of edges
D3 decomposing graphs according to labels of vertices and then decomposing them according to labels of edges
D4 decomposing graphs according to labels of edges and then decomposing them according to labels of vertices

For decomposed graphs according to labels of both vertices and edges, we also consider the following ways to use their eigenvalues.

E1 using eigenvalues of only graphs decomposed completely according to labels of both vertices and edges
E2 using not only the above eigenvalues but also eigenvalues of intermediate graphs decomposed according to only labels of vertices or edges

6. Experimental Evaluation

We evaluate the four combinations of decomposing graphs and using the eigenvalues, that is, D3 and E1, D3 and E2, D4 and E1, and D4 and E2 in the above section. We denote the algorithms based on these combinations as VES, VEA, EVS and EVA, respectively. We compare them with the algorithm without graph decomposition denoted as NoDecomp, which was proposed by Nagaya et al. [1] and VF2. VF2 is one of the state-of-the-art algorithms for deciding whether a graph contains another graph, which uses only combinatorial methods. We use undirected labeled graphs which are randomly generated with the graph generator which we prepare. All algorithms are implemented with Microsoft Visual C++ 2010. For tridiagonalization of symmetric martices, we call Matlab R2009a from the programs. Since VF2 is originally implemented for checking subgraph isomorphism of directed labeled graphs, we customize it for undirected labeled graphs and compile it with Microsoft Visual C++ 2010. All experiments done on a PC running Microsoft Windows Vista Business 64 bit SP2 with an Intel Xeon E5420 processor and 16 GB RAM. We use 500 pairs of graphs and the larger graph of the pair has 100 vertices and 2000 edges in every experiment and repeat each experiment 10 times.

6.1 Processing Time

We show the processing time of the proposed methods, the previous method NoDecomp and VF2 in Tables 1 and 3. For the proposed methods, it is the average time required to check whether a graph is not a subgraph of another graph and, if it might be, for VF2 to confirm whether the graph is surely a subgraph. For VF2, it is the average time required to check whether the graph is a subgraph. In Table 1, the smaller graph of the pair has 3 labels each for vertices and edges and the density is 0.45. The number of vertices vary from 40 to 90. All proposed methods are considerably faster than NoDecomp. VES and EVS of the proposed methods

Table 1 Processing time in varying number of vertices of smaller graph [sec].

Vertices	40	50	60	70	80	90
VEA	137.8	128.0	80.4	32.8	5.9	0.5
VES	119.0	**106.9**	**62.8**	19.4	**1.9**	**0.4**
EVA	385.2	447.3	425.3	278.9	74.4	0.6
EVS	118.5	108.1	**62.8**	**19.2**	2.0	**0.4**
NoDecomp	2103.1	2207.0	2071.0	1508.3	604.5	2.9
VF2	**113.9**	123.7	109.0	58.9	37.8	6.4

Table 2 Number of detected non-subgraphs in varying number of vertices of smaller graph.

Vertices	40	50	60	70	80	90
VEA	**39.5**	**113.3**	**280.7**	446.5	498.5	**500.0**
VES	**39.5**	113.2	280.3	446.5	498.5	**500.0**
EVA	**39.5**	113.2	280.3	**446.9**	**498.8**	**500.0**
EVS	**39.5**	113.2	280.3	446.5	498.5	**500.0**
NoDecomp	4.6	21.2	84.5	230.5	420.7	499.9

Table 3 Processing time in varying number of labels of pair of graphs [sec].

Labels	5	6	7	8	9	10
VEA	500.3	**26.4**	3.5	1.0	0.6	0.5
VES	497.0	28.7	3.3	**0.8**	**0.5**	**0.4**
EVA	886.8	253.2	115.1	47.9	14.9	5.4
EVS	**496.2**	28.7	**3.1**	**0.8**	**0.5**	**0.4**
NoDecomp	2492.3	1019.3	522.1	211.8	70.3	23.9
VF2	1429.8	208.7	53.8	13.8	5.5	2.5

have almost the same performance and are faster than VEA and EVA. EVA is the slowest in the proposed methods. VES and EVS are faster than VF2 except the case of 40 vertices. The proposed methods become faster than VF2, as the number of vertices of the smaller graph increases. In Table 3, the smaller graph of the pair has 70 vertices, 700 edges and 2 labels of edges. The number of labels of vertices of the pair of graphs vary from 5 to 10. All proposed methods are considerably faster than NoDecomp again. VEA, VES and EVS of the proposed method have almost the same performance and are faster than EVA. VEA, VES and EVS are always faster than VF2 in this experiment. The proposed methods become faster than VF2, as the number of labels of vertices of the pair increases.

6.2 Effectiveness in Detecting Non-subgraphs

We evaluate the performance of the proposed methods in detecting whether a graph is not a subgraph of another graph. In Tables 2 and 4, we use the same set of graphs as in the experiments of Tables 1 and 3, respectively. For all of the 500 pairs of graphs, the smaller graph is not a subgraph of the larger graph. The tables show the number of pairs detected that the smaller graph is not a subgraph of the larger graph by the proposed methods and NoDecomp. Table 2 and 4 shows that the performance of the proposed methods degrades less than NoDecomp, as the number of labels of vertices of the pair decreases. Although VES and EVS use

Table 4 Number of detected non-subgraphs in varying number of labels of pair of graphs.

Labels	5	6	7	8	9	10
VEA	**359.7**	**449.3**	**479.4**	491.4	**497.8**	**499.5**
VES	359.1	448.0	478.6	491.1	492.3	487.7
EVA	359.1	448.1	479.0	**492.0**	494.7	492.1
EVS	359.1	448.0	478.6	491.1	492.3	487.4
NoDecomp	204.0	319.3	399.0	457.6	485.9	495.0

fewer eigenvalues than VEA and EVA, VES and EVS have almost the same performance as VEA and EVA.

7. Conclusion

We propose a method to detect non-subgraphs more efficiently by comparing the eigenvalues of graphs decomposed according to labels of the vertices and the edges. Our method is based on the interlacing property of eigenvalues between a graph and its subgraph. The experimental evaluation shows that the proposed method reduces the cost of computing eigenvalues and increases the chance of detecting non-subgraphs in comparison with our previous method.

Acknowledgments

The authors are grateful to the anonymous reviewer for helpful comments. This work was supported by JSPS Grant-in-Aid for Scientific Research (C) Grant Number 21500104.

References

[1] H. Nagaya, K. Katayama, and H. Ishikawa, "Application of Cauchy's interlace theorem to subgraph isomorphism detection for large graphs," IEICE 17th Data Engineering Workshop (DEWS2006), 3B-i9, 2006.

[2] W.H. Haemers, "Interlacing eigenvalues and graphs," Linear Algebra Appl., vol.226, pp.593–616, 1995.

[3] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento, "An improved algorithm for matching large graphs," 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition, pp.149–159, 2001.

[4] J.R. Ullmann, "An algorithm for subgraph isomorphism," J. Assoc. Comput. Mach, vol.23, no.1, pp.31–42, 1976.

[5] B.D. McKay, "Practical graph isomorphism," Congressus Numeranitum, vol.30, pp.45–87, 1981.

[6] A. Shokoufandeh, S.J. Dickinson, K. Siddiqi, and S.W. Zucker, "Indexing using a spectral encoding of topological structure," 1999 Conference on Computer Vision and Pattern Recognition, pp.2491–2497, 1999.

[7] N. Zhang, M.T. ´Ozsu, I.F. Ilyas, and A. Aboulnaga, "FIX: Feature-based indexing technique for XML documents," 32nd International Conference on Very Large Data Bases, VLDB Endowment, pp.259–270, 2006.

[8] L. Zou, L. Chen, J.X. Yu, and Y. Lu, "A novel spectral coding in a large graph database," 11th International Conference on Extending Database Technology, pp.181–192, 2008.

[9] N. Vinh, A. Inokuchi, and T. Washio, "Graph classification based on optimizing graph spectra," Discovery Science, Lect. Notes Comput. Sci., vol.6332, pp.205–220, Springer Berlin / Heidelberg, 2010.