A Critical Examination of Node-Similarity Based Graph Matching Algorithms

Guoxing Zhao[1], Miltos Petridis[1], Grigori Sidorov[2] and Jixin Ma[1]

[1]School of Computing and Mathematical Sciences,
the University of Greenwich, U.K.
[2]Center for Research in Computer Science,
National Polytechnic Institute, Mexico
j.ma@gre.ac.uk

Abstract. In this paper, we shall critically examine a special class of graph matching algorithms that follow the approach of node-similarity measurement. A high-level algorithm framework, namely node-similarity graph matching framework (NSGM framework), is proposed, from which, many existing graph matching algorithms can be subsumed, including the eigen-decomposition method of Umeyama, the polynomial-transformation method of Almohamad, the hubs and authorities method of Kleinberg, and the kronecker product successive projection methods of Wyk, etc. In addition, improved algorithms can be developed from the NSGM framework with respects to the corresponding results in graph theory. As the observation, it is pointed out that, in general, any algorithm which can be subsumed from NSGM framework fails to work well for graphs with non-trivial auto-isomorphism structure.

Keywords: Graph matching, node-similarity.

1 Introduction

Graph, as a powerful and versatile mathematical tool, is widely used for the description of structural objects in many application areas such as case-based reasoning, semantic networks, document processing, image analysis, biometric identification, computer vision, video analysis, and so on.

In applications such as pattern recognition and computer vision, object similarity becomes the most important issue and based on the graph representation, object similarity is simply transfer into the similarity degree between two graphs which is known as the graph matching problems.

Various algorithms for graph matching problems have been developed, which, according to [7], can be classified into two categories: (1) search-based

Advances in Artificial Intelligence: Algorithms and Applications
Research in Computing Science 40, 2008, pp. 73-82

methods which rely on possible and impossible pairings between vertices; and (2) optimization-based methods which formulate the graph matching problem as an optimization problem. Generally speaking, on one hand, search-based methods will find optimal solutions, but require exponential time in the worst case. On the other hand, optimization-based methods normally require only polynomial-bounded computational time, but in some cases may fail to find the optimal solution.

Most search-based approaches use the idea of heuristics [13, 15, 16] to reduce the size of the searching space, while optimization-based methods have followed distinct approaches, which again can be roughly classified in to two groups, (1) traditional optimization based methods including linear programming methods [2], quadratic programming approaches [12], Bayesian methods [6], relaxation labeling [9], neural network [14], genetic algorithm [11] and so on; (2) special theory based methods including symmetric polynomials transformation [1], kronecker product successive projection [4], eigen-decomposition method [17, 18], spectral embedding approach [3], and hubs and authorities method [5, 10], etc. In general, the traditional optimization based methods transfer the graph matching problem to a classical optimization problem and traditional optimization algorithm can be directly applied, but this transformation sometimes conceals the essence and makes the algorithm hard to be analyzed and improved. On the other hand, every special theory based method usually provides a simple theoretical foundation to deduce and analyze the corresponding graph matching algorithm, so the principles and applied scope is relatively clearer. However, most special theory based methods are only applicable for very special kind of graph pairs.

In this paper, instead of a new graph matching algorithm, an abstract algorithm template called node-similarity graph matching algorithm template shall be presented. This algorithm template can be seen as an abstraction of many graph matching algorithms, such as SPGM of Almohamad [1], EDGM of Umeyama [17], HAGM of Kleinberg [10], LSKPGM of Wyk [4], etc. In this sense, a unified analysis and comparison can be provided on the starting points, execution processes and constraints of these graph matching algorithms. The rest of this paper is organized as following: Basic notations of graph matching problem and node-similarity matching algorithm template are introduced in section 2. Several concrete node-similarity algorithms are proposed, tested and compared in section 3. In section 4 we shall extends the node-similarity graph matching algorithm template for matching multiple-weighted graph pairs and in section 5 an important limitation of node-similarity graph matching algorithms are pointed out that all these algorithms are not suitable for self-similar graphs. Section 6 simply concludes this paper.

2 Graph Matching Problems

Following the definition of [1], a weighted graph G is denoted as an ordered pair (N, w), where N is a set of nodes and w is a weighting function assigning a weight $w(v_i, v_j)$ to each pair of nodes (v_i, v_j) (edge of the graph). The adjacency matrix of a weighted graph G = (V, w) is defined as $A_G = \{g_{ij}\}$, where $g_{ij} = w(v_i, v_j)$, i, j = 1, 2, ..., n, and n is the number of nodes in graph G.

Note: in this paper, only fully weighted graphs with no multiple edges are considered.

For the reason of simplifying repression, without confusion, we shall not distinguish a weighted graph G and its corresponding adjacency matrix A_G. In other words, we shall simply express the adjacency matrix of G as graph G itself.

The problem of matching two weighted graphs G and H of n nodes is to find a one-to-one correspondence between the two corresponding sets of nodes that minimizes the distance between G and H, which can be formulated in terms of Frobenius-norm (denoted as $\|\bullet\|_F$) as:

$$\arg\min_{P\in perm(n)} \|PGP^T - H\|_F \tag{1}$$

Where G and H are the adjacent matrices of the weighted graphs to be matched and perm(n) is the set of all n-by-n permutation matrices.

Note: in this paper, we only handle the matching of two graphs with the same size.

3 Node-similarity Graph Matching Algorithm Template

In general, completely solving formula (1) is nearly impossible, so we propose the node-similarity based algorithm template as follow:

$$\arg\min_{P\in perm(n)} \|P - S(G,H)\|_F \tag{2}$$

Where S is a node-similarity function from $R^{n\times n} \times R^{n\times n}$ to $R^{n\times n}$ satisfying

$$S(PGP^T, H) = S(G,H)P^T, \text{ for all } P \in perm(n) \tag{3}$$

In other words, S(G, H) is a n-by-n matrix whose (i,j)-entry expressing some kinds of similarity between j-th node of graph G and i-th node of graph H. (DO NOT mistake the order of i and j here)

Note 1: formula (2) is called a template either than an algorithm, because the function S is left to be open, and each implementation of S will provide corresponding node-similarity, and then matching algorithm.

Note 2: formula (3) states that the similarity of two nodes is independent with the order of the node in the graph.

Note 3: formula (2) can be easily solved by the Hungarian algorithm [8].

So a unified framework to generate and analyze graph matching algorithm is constructed based on the implementation of node-similarity function S.

4 Some Algorithms of NSGM Template

In this section, some algorithms of the NSGM template will be introduced. We use several Matlab command as denotations that supposing T is any symbols expressing a matrix, then T(k,:) denotes the k-th row of T, G(:,k) denotes k-th column, and T(i,j) denotes the ij-th entry of T. $\otimes$ denotes the kronecker product operator and vec() denotes the vectorization operator. sum(), sort(), poly(), and diag() have the same meaning with corresponding Matlab functions.

4.1 Directly Constructing NSGM Algorithms

Theorem 1: The following functions are all node-similarity functions.

$S_1(G,H)(i,j) = -|G(j,j) - H(i,i)|$

$S_2(G,H)(i,j) = \sum_{k=1}^{n}\sum_{l=1}^{n} G(k,j) \times H(l,i)$

$S_3 = \mathrm{vec}((G \otimes H + G^T \otimes H^T)^m \times \mathbf{1}_{n^2 \times 1})$, where m can be any natural number and $\mathbf{1}_{n^2 \times 1}$ is a column vector with all n^2 elements 1.

The proof can be provided by verifying that formula (3) holds for S_1, S_2 and S_3, where 1) and 2) is elementary and 3) can be simply proved by induction on m. Details are omitted here.

Note 1: the corresponding node-similarity matching algorithm of function S_2 is actually equivalent to the least square kronecker product graph matching algorithm of Wyk [4].

Note 2: the corresponding node-similarity matching algorithm of function S_3 is actually equivalent to the hubs and authorities matching algorithm of Kleinberg [10].

Obviously, LSKPGM and HAGM algorithms have been re-interpreted in a simple and unified form.

4.2 Constructing NSGM Algorithms by Node-attribute Functions

From Theorem 1, it can be seen that to construct a node-similarity function, one has to consider a graph pair G and H at the same time, which sometimes

makes the construction a little complicated. In this section a new kind of functions called node-attribute functions will be introduced to simplify the construction.

Definition 1: A node-attribute function is a function $f: R^{n\times n} \to R^{n\times m}$ which satisfies

$$f(PGP^T) = Pf(G), \text{ for all } G \in R^{n\times n} \text{ and } P \in \text{perm}(n) \tag{4}$$

Intuitively, f can be seen as a function mapping the edge attribute of graph G to its node attribute.

We shall give some examples of the node-attribute functions.

Theorem 2 : The following functions are all node-attribute functions:

$f_4(G)(k,:)=[\text{sum}(G(k,:)), \text{sum}(G(:,k))]$
$f_5(G)(k,:)=[\text{sort}(G(k,:)), \text{sort}(G(:,k))]$
$f_6(G)(k,:)=[\text{poly}(G(k,:)), \text{poly}(G(:,k))]$

$$f_7(G)(:,k)=\text{diag}\left(\frac{G^k}{n^k}\right), \quad k=1,2,\ldots,n.$$

The proof can be provided by verifying that formula (4) holds for f_4-f_7, which are elementary and are omitted here.

Theorem 3: Function f_8 satisfies formula (4) if the matrix G' (defined below) only has single eigenvalues.

$f_8(G)=A$, where A is calculated by 3 steps:

8.1) $G' := \frac{G+G^T}{2} + \frac{G-G^T}{2}\sqrt{-1}$

8.2) Calculating the eigen-decomposition of the matrix $G'=UDU^*$, where D is the diagonal matrix of eigenvalues in descending order.

8.3) $A = \text{abs}(U)$, which is the matrix whose entries are the absolute value of corresponding entries of U.

The proof is trivial.

Note: the function f_8 doesn't satisfy formula (4) for general case if the matrix G' has multiple eigenvalues.

Using the node-attribute function, one can easily construct the corresponding node-similarity function by the following theorem.

Theorem 4: Let f be a node-attribute function, and S_f is defined as:

$$S_f(G,H) = f(H) \times f(G)^T$$

Then S_f is a node-similarity function.

Followed by Theorem 4, each node-attribute function in this Theorem defines a corresponding node-similarity function denoted as S_4, to S_8. The matching algorithm by node-similarity functions S_6 and S_8 are exactly the symmetric polynomials transformation graph matching algorithm SPGM of Almohamad [1] and the eigen-decomposition method EDGM of Umeyama [17] respectively.

4.3 Testing of NSGM Algorithms

In this section, these algorithms generated by node-similarity function from S_1 to S_8 are numerically compared. In each test, 500 pairs of isomorphic matrices are generated by Matlab, whose elements are uniformly random numbers in [0, 1]. For each pair G and H, a perturbation matrix E is added to Graph H, where every elements of E is a uniformly random number in $[0, \varepsilon]$.

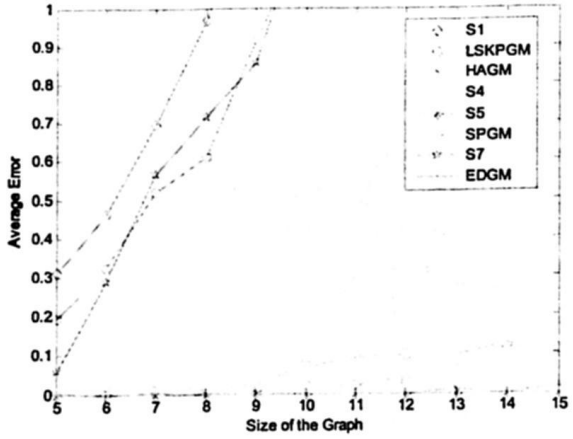

Fig. 1. Average error for ε =0.05.

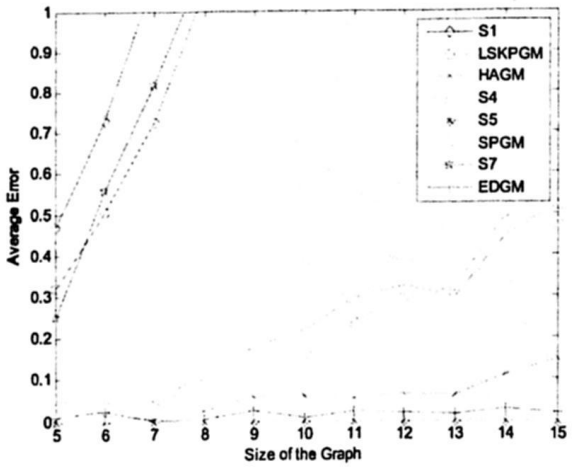

Fig. 2. Average error for ε =0.10

Then the average of matching error for these algorithms are calculated and compared. The matching error is defined as:

$$er(S,G,H) = \left\| P_S G P_S^T - H \right\|_F - \left\| P_0 G P_0^T - H \right\|_F$$

where P_0 is the best matching permutation of G and H, P_S is the matching result calculated by node-similarity algorithm corresponding to node-similarity function S.

The test result for ε =0.05, 0.10 and 0.15 are shown in Fig 1-Fig 3.

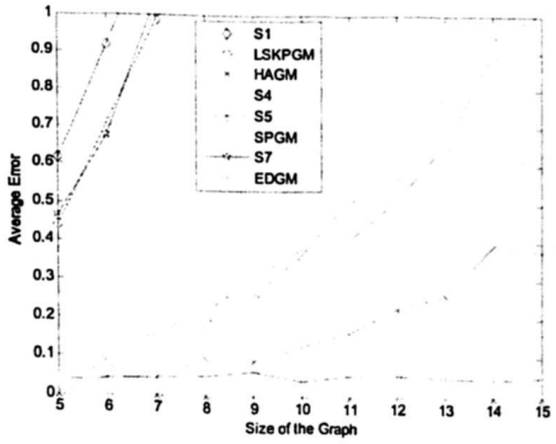

Fig. 3. Average error for ε =0.15

These eight algorithms are clearly in three groups: (1) inaccurate algorithms including S_1, S_7 and LSKPGM; (2) roughly accurate algorithms including S_4 and SPGM; (3) accurate algorithms including: HAGM, EDGM and S_5. The algorithm of S_5 based on simply sorting the matrix elements, gets almost zero-error in all three tests, which in one respect indicates that complex computations usually cover the essence of the problem instead of revealing the essence.

4.4 Computational Complexity

All these algorithms have two main steps, the calculation of node-similarity and the calculation of permutation by Hungarian algorithm. The latter's computational is well known as $O(n^3)$, where n is the number of nodes in graph G and H.

Table 1. Computational complexity of NSGM algorithms

Complexity / Algorithm	Calculating node-similarity	Calculating permutation matrix	In All
S1	$O(n^2)$	$O(n^3)$	$O(n^3)$
LKPGM	$O(n^4)$	$O(n^3)$	$O(n^4)$
HAGM	$O(n^3m)$	$O(n^3)$	$O(n^3m)$
S4	$O(n^3)$	$O(n^3)$	$O(n^3)$
S5	$O(n^3)$	$O(n^3)$	$O(n^3)$
SPGM	$O(n^3)$	$O(n^3)$	$O(n^3)$
S7	$O(n^4)$	$O(n^3)$	$O(n^4)$
EDGM	$O(n^3)$	$O(n^3)$	$O(n^3)$

5 Extension

In the discussion above, only the weighted graphs are considered. But at some circumstance, we have to deal with the multi-weighted graph matching problems which can be defined as:

$$\arg\min_{P\in perm(n)} \sum_{i=1}^{m} \left\| PG_iP^T - H_i \right\|_F$$

where the G_i and H_i are the i-th weight matrices of graph G and H.

The node-similarity algorithm template can be directly expanded for these multi-weighted graphs as:

$$\arg\min_{P\in perm(n)} \left\| P[I,I,\cdots,I] - [S(G_1,H_1),S(G_2,H_2),\cdots,S(G_m,H_m)] \right\|_F$$

where I is the n-by-n identity matrix.

From above, we can see that to solve a multi-weighted graph matching problem using the node-similarity algorithm is as simple as the normal weighted graph matching problem.

6 Limitation of NSGMT

Let S be a node-similarity function satisfying (3)

$S(PGP^T, H) = S(G, H)P^T$, for all $P \in perm(n)$.

Specially, we choose $P_0 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$

Graph G is also chosen to be special that satisfying

$$P_0 G P_0^T = G \tag{5}$$

Then from formula (3), we get

$$S(G,H) = S(P_0 G P_0^T, H) = S(G,H) P_0^T \tag{6}$$

Which means all columns of S(G, H) are equal

$$S(G,H) = [v, v, \cdots v] \tag{7}$$

Then we get the following theorem:

Theorem 5: Let graph G be a circle, then G satisfies formula (4) and (6). In this case, any permutation is an optimal solution of formula (2). The node-similarity algorithm fails to work.

Theorem 5 only claims that NSGMT are not applicable for circles, how about others? In fact, the essence of this failure is that if graph G is self-similar, which means there is a non-trivial automorphism of graph G, the NDGMT will fail to distinguish those similar nodes of G.

7 Conclusion

In this paper, a unified framework for generating, testing, analyzing, comparing and expanding the node-similarity graph matching algorithms are presented. On one hand, this work provides a new view on those traditional graph matching algorithms based on different theories to see them consistently. On the other hand, this work also point out the essential limitation of solving graph matching problems by simply comparisons of node similarity. This gives us an important enlightment for developing new matching algorithms beyond this algorithm template.

References

1. Almohamad, H.A.: A Polynomial Transform for Matching Pairs of Weighted Graphs. Applied Mathematical Modelling, Vol. 15, pp. 216-222. Elsevier Science (1991)

2. Almohamad, H.A. and Duffuaa, S.O.: A Linear Programming Approach for the Weighted Graph Matching Problem. IEEE Trans. PAMI, Vol. 15, pp. 522-525 (1993)
3. Bai, X., Yu, H. and Hancock, E.R.: Graph matching using spectral embedding and alignment. Pattern Recognition, Vol. 3, Issue 23-26, pp. 398 – 401 (2004)
4. Barend Jacobus van, Wyk.: Kronecker Product Successive Projection and Related Graph Matching Algorithms. Ph.D. diss., University of the Witwatersrand, Johannesburg (2002)
5. Blondel, V., Gajardo, A., Heymans, M., Senellart, P. and Van Dooren, P.: A measure of similarity between graph vertices: applications to Synonym Extraction and Web Searching. SIAM Rev., Vol. 46, No. 4, pp. 647-666 (2004)
6. Finch, A.M., Wilson R.C. and Hancock, E.R.: Matching Delaunay Triangulations by Probabilistic Relaxation. Proc. of Computer Analysis of Images and Patterns, pp. 350-358 (1995)
7. Gold, S. and Rangarajan, A.: A Graduated Assignment Algorithm for Graph Matching. IEEE Trans. PAMI, Vol. 18, pp. 377-388 (1996)
8. Harold, W.K.: The Hungarian Method for the Assignment Problem. Naval Research Logistic Quarterly, Vol. 2, pp. 83-97 (1995)
9. Hummel, R. and Zuker, S.: On the Foundations of Relaxation Labeling Processes. IEEE Trans. PAMI 5, pp. 267-287 (1983)
10. Kleinberg, J.: Authoritative sources in a hyperlinked environment. Journal of the ACM, Vol. 46, No. 5, pp. 604-632 (1999)
11. Kremar, M. and Dhawan, A.P.: Application of Genetic Algorithms in Graph Matching. Proc. of the International Conference on Neural Networks 6, pp. 3872-3876 (1994)
12. Neuhaus, M. amd Bunke, H.: A Quadratic Programming Approach to the Graph Edit Distance Problem. Journal of Lecture Notes in Computer Science, Vol. 4538, pp. 92-102 (2007)
13. Sanfeliu, A. and Fu, K.S.: A Distance Measure between Attributed Relational Graphs for Pattern Recognition. IEEE Trans. SMC 13, pp. 53-363 (1983)
14. Suganthan, P., Teoh, E. and Mital, D.: Pattern Recognition by Graph Matching Using the Potts MFT Neural Networks. Pattern Recognition 28, pp. 997-1009 (1995)
15. Tasi, W.H. and Fu, K.S.: Error-Correcting Isomorphisms of Attributed Relational Graphs for Pattern Recognition. IEEE Trans. SMC 9, pp.757-768 (1979)
16. Ullman, J.R.: An Algorithm for Subgraph Isomorphism. Journal of the Association for Computing Machinery Vol. 23, pp. 31-42 (1976)
17. Umeyama, S.: An Eigendecomposition Approach to Weighted Graph Matching Problems. IEEE Trans. PAMI, Vol. 10, pp. 695-703 (1988)
18. Zhao, G., Luo, B., Tang J. and Ma, J.: Using Eigen-Decomposition Method for Weighted Graph Matching. Lecture Notes in Computer Science Vol. 4681, pp. 1283-1294 (2007)