

Graph similarity scoring and matching

Laura A. Zager* and George C. Verghese

April 10, 2007

Abstract

We outline a class of graph similarity measures that use the structural similarity of local neighborhoods to derive pairwise similarity scores between the nodes of two different graphs, and present a related similarity measure that uses a linear update to generate both node and edge similarity scores. This measure is then applied to the task of graph matching.

Keywords: graphs and networks, graph algorithms, similarity measures, graph matching, graph alignment

1 Introduction

Many applications call for a quantitative measure of the ‘similarity’ of two graphs. A good deal of research has been devoted to the graph isomorphism problem and to its generalizations, edit distance and maximum (minimum) common sub(super)graph (see, e.g. [1] - [5]). Additionally, the emergence of very large graphs like the World Wide Web has generated tremendous interest in aggregate statistical measures of graph structure, such as clustering, diameter, and degree distribution. Between these two ends of the spectrum, there exists a class of similarity methods in which an element (e.g., a node or edge) in graph G_A and an element in graph G_B are considered similar if their respective neighborhoods within G_A and G_B are similar. This idea naturally leads to iterative methods for computing similarity scores between the elements of these graphs, in which scores between elements propagate along to neighboring elements at each time step. Here, we consider a graph $G_A(V_A, E_A)$ to consist of a set of *vertices* or *nodes* V_A , and a set of *edges* $E_A \subseteq V_A \times V_A$, which can be directed or undirected. The aim of this paper is to highlight graph similarity methods from different fields that utilize this iterative framework, and present a simple application of one particular extension to the task of graph matching.

One influential iterative approach was the HITS search algorithm introduced by Kleinberg in [6]. He suggested that useful web search results could be divided into two different categories: ‘hubs,’ which point to many good sources of information on the query; and ‘authorities,’ which are pointed to by high-quality hubs. Kleinberg proposed an iterative algorithm for computing a hub and authority score for each node in the web graph G_W ; the authority score of a node n at iteration k is simply the sum of the hub scores of the nodes that point to node n at iteration $k - 1$, and the hub score of node n at iteration k is the sum of the authority scores of the nodes to which node n points. The even and odd iterates of this procedure will each converge, but possibly to different limits which may depend on the initial values chosen.

Several other application-oriented algorithms also utilize the idea of recursively computing node similarity scores based on the scores of neighboring nodes; these are the *similarity flooding* algorithm proposed by Melnik, Garcia-Molina and Rahm [7], the *SimRank* algorithm proposed by Jeh and Widom [8], the scoring method for phylogenetic tree construction proposed by Heymans and Singh [9], and the vertex similarity method of Leicht et al. [10].

In [11], Blondel et al. view Kleinberg’s algorithm as a comparison between each node in the graph G_W to the two nodes of a smaller ‘prototype’ *hub-authority graph* G_{HA} , depicted in Figure 1. The hub and

*Corresponding author. E-mail: lzager@mit.edu. Postal address: MIT, Rm 10-082, 77 Massachusetts Ave., Cambridge, MA 02139.

authority scores of each node in G_W are seen as similarity measures between each node in G_W , and the hub and authority nodes in G_{HA} , respectively.

In light of this observation, Blondel et al. generalize Kleinberg's update equation to construct a similarity measure between *any* two nodes in *any* two graphs. Blondel et al. denote the similarity of node i in G_B and node j in G_A at stage k by $x_{ij}(k)$ and define it via the following iteration:

$$\tilde{x}_{ij}(k) = \sum_{r:(r,i) \in E_B, s:(s,j) \in E_A} x_{rs}(k-1) + \sum_{r:(i,r) \in E_B, s:(j,s) \in E_A} x_{rs}(k-1).$$

This update is followed by a normalization of all of the scores by $\sum_{i,j} \tilde{x}_{ij}^2$, which we will denote by $x_{ij} \leftarrow \tilde{x}_{ij}$. As with the HITS algorithm, this procedure will, in general, converge to different even and odd limits which will depend upon the initial conditions. Blondel et al. choose to initialize x_0 to the all-ones vector, and choose their final score to be the limit of the *even* iterations.

We next present a related similarity measure that uses a linear update to generate both node and edge similarity scores that do not depend on the initial values of the node scores. We then explore the application of this measure to graph matching.

2 Coupled node-edge scoring

Using the iterative method framework, in which two graph elements are similar if their neighborhoods are similar, we can immediately suggest one simple way to construct an edge score: *an edge in G_B is like an edge in G_A if their respective source and terminal nodes are similar*. This definition of edge similarity introduces a coupling between edge and node scores.

Let $s_A(i)$ denote the source node of edge i in graph G_A , and let $t_A(i)$ denote the terminal node of edge i . The node set cardinality $|V_A|$ will be denoted by n_A and its edge set cardinality $|E_A|$ by m_A . It will be most convenient to represent the adjacency structure G_A by a pair of $n_A \times m_A$ matrices, the *source-edge matrix* A_S and the *terminus-edge matrix* A_T , defined as follows:

$$[A_S]_{ij} = \begin{cases} 1 & s_A(j) = i \\ 0 & \text{otherwise} \end{cases}$$

$$[A_T]_{ij} = \begin{cases} 1 & t_A(j) = i \\ 0 & \text{otherwise} \end{cases}$$

This representation has some useful properties:

- $D_{A_S} \equiv A_S A_S^\top$ is a diagonal matrix with the *out-degree* of node i in the i th diagonal entry.
- $D_{A_T} \equiv A_T A_T^\top$ is diagonal with the *in-degree* of each node in the i th diagonal entry.

Let x_{ij} denote the node similarity score between node i in G_B and node j in G_A , and y_{pq} the edge score between edge p in G_B and edge q in G_A . An update equation for edge and node scores takes the following form:

$$y_{pq}(k) \leftarrow x_{s(p)s(q)}(k-1) + x_{t(p)t(q)}(k-1) \quad (1)$$

$$x_{ij}(k) \leftarrow \sum_{t(k)=i, t(l)=j} y_{kl}(k-1) + \sum_{s(k)=i, s(l)=j} y_{kl}(k-1). \quad (2)$$

These scores can be assembled into matrices: Y_k and X_k . A normalization at each stage by the Frobenius norm (or any appropriate matrix norm) is required, as indicated by the \leftarrow notation.

Using the construction of source-edge matrices A_S and terminus-edge matrices A_T of a graph G_A , the update process represented in Eqs. 1 and 2 can be written concisely in the following matrix form:

$$Y_k \leftarrow B_S^\top X_{k-1} A_S + B_T^\top X_{k-1} A_T, X_k \leftarrow B_S Y_{k-1} A_S^\top + B_T Y_{k-1} A_T^\top.$$

It is useful to introduce the $\text{vec}(\cdot)$ operator on matrices, which stacks the columns of a matrix into a vector as follows:

$$\text{vec} \left(\begin{bmatrix} | & | & \cdots & | \\ v_1 & v_2 & \vdots & v_m \\ | & | & \cdots & | \end{bmatrix} \right) = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}.$$

Define $y_k = \text{vec}(Y_k)$ and $x_k = \text{vec}(X_k)$. Then the update can be expressed as:

$$y_k \leftarrow (A_S^\top \otimes B_S^\top + A_T^\top \otimes B_T^\top) x_{k-1} \equiv G x_{k-1} \quad (3)$$

$$x_k \leftarrow (A_S \otimes B_S + A_T \otimes B_T) y_{k-1} \equiv G^\top y_{k-1} \quad (4)$$

where \otimes denotes the Kronecker product of matrices. With this representation, determining the score vector x to which this procedure converges is a matrix iteration problem. Concatenating these vectors yields a single matrix update equation:

$$s_k \equiv \begin{bmatrix} x \\ y \end{bmatrix}_k \leftarrow \begin{bmatrix} 0 & G^\top \\ G & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}_{k-1} \equiv M s_{k-1}. \quad (5)$$

The following theorem summarizes some of the key properties of this graph similarity measure s_k .

Theorem 2.1 *With the initial condition x_0 chosen arbitrarily and with $y_0 = \alpha G x_0$, where $\alpha > 0$ is any positive constant, the iteration of Eq. 5 will converge to a unique set x of nonnegative similarity scores between every pair of nodes in G_A and G_B and a corresponding unique set y of nonnegative similarity scores between every pair of edges.*

Proof: First, we note the following result, presented as Theorem 2 in [11]. If A is a symmetric, non-negative matrix with Perron root ρ , and b_0 is a component-wise positive vector of appropriate dimension, define:

$$b_{k+1} = \frac{A b_k}{\|A b_k\|_2}, k = 0, 1, \dots$$

Then, provided $-\rho$ is not an eigenvalue of A , the sequence b_k converges to $\Pi b_0 / \|\Pi b_0\|_2$, where Π is the orthogonal projector onto the invariant subspace associated with ρ .

To apply this theorem directly, we can expand the matrix equation given in Eq. 5 into explicit expressions for x_k and y_k in terms of the initial conditions:

$$x_k \leftarrow \begin{cases} (G^\top G)^{k/2} x_0 & k \text{ even} \\ (G^\top G)^{(k-1)/2} G^\top y_0 & k \text{ odd} \end{cases}$$

$$y_k \leftarrow \begin{cases} (G G^\top)^{k/2} y_0 & k \text{ even} \\ (G G^\top)^{(k-1)/2} G x_0 & k \text{ odd} \end{cases}$$

Applying the requirement that $\alpha G x_0 = y_0$ to the expressions for x_k and y_k :

$$x_k \leftarrow \begin{cases} (G^\top G)^{(k-2)/2} G^\top G x_0 = \frac{1}{\alpha} (G^\top G)^{(k-2)/2} G^\top y_0 & k \text{ even} \\ (G^\top G)^{(k-1)/2} G^\top y_0 & k \text{ odd} \end{cases}$$

$$y_k \leftarrow \begin{cases} (G G^\top)^{k/2} y_0 & k \text{ even} \\ (G G^\top)^{(k-1)/2} G x_0 = (G G^\top)^{(k-1)/2} \frac{1}{\alpha} y_0 & k \text{ odd} \end{cases}$$

Observe that the matrices GG^\top and $G^\top G$, besides being symmetric and non-negative, are also positive semi-definite, and thus have only nonnegative eigenvalues. Therefore, the result presented in Theorem 2 of [11] applies. Note that the factor of $\frac{1}{\alpha}$ that appears in the odd iterates will be eliminated by the normalization embedded in each step. In the limit as $k \rightarrow \infty$, the even and odd expressions are equal for both x and y . ■

The matrix G as defined in Eq. 3 is the sum of two matrices, each of which have a single ‘1’ entry in each row. Thus, if x_0 is chosen to be the all-ones vector, a reasonable choice if no *a priori* information about node similarity is known, then $Gx_0 = 2y_0$ where y_0 is also an all-ones vector, and the initial condition constraint is satisfied. An example of the application of this method to the pair of graphs in Figure 2 is presented in Table 1.

Rather than a *simultaneous* update of node and edge scores, consider a *sequential* update, which could be defined as follows:

$$y_k = Gx_{k-1}, x_k = G^\top y_k \quad (6)$$

or as:

$$x_k = G^\top y_{k-1}, y_k = Gx_k.$$

It is not difficult to show, by an analogous argument to the proof of Theorem 2, that the simultaneous and sequential update procedures all yield the same set of similarity scores.

Since the even and odd limits of the iteration converge to the same scores, we can focus on the even iterations and expand the expression for the node similarity update:

$$\begin{aligned} x_k &\leftarrow (G^\top G)x_{k-2} \\ &= (\mathcal{A} \otimes \mathcal{B} + \mathcal{A}^\top \otimes \mathcal{B}^\top + D_{A_S} \otimes D_{B_S} + D_{A_T} \otimes D_{B_T})x_{k-2} \end{aligned}$$

where $\mathcal{A} = A_S A_T^\top$ and $\mathcal{B} = B_S B_T^\top$ are the standard node-node adjacency matrices of G_A and G_B and the matrices D_{A_S} and D_{A_T} are defined in Section 2. Note that the iteration in [11] has the form:

$$x_k \leftarrow (\mathcal{A} \otimes \mathcal{B} + \mathcal{A}^\top \otimes \mathcal{B}^\top)x_{k-1}.$$

Thus, the coupled node calculation differs from that in [11] by the inclusion of additional diagonal terms that serve to amplify the scores of nodes that are highly connected.

3 Application: graph matching

One common task in graph theory applications is the identification of some kind of optimal *matching* between the respective elements (i.e., nodes and edges) of two graphs. Graph matching has received an enormous amount of academic treatment, in the pattern matching and data mining communities in particular, and many computationally-efficient methods exist for different classes of graphs. What is often sought in these problems is an *assignment matrix* P ; if set B has m elements and set A has $n \geq m$ elements, then P will be an $m \times n$ matrix of 0’s and 1’s, with a single 1 entry on each row, and no more than a single 1 entry in each column. If $P_{ij} = 1$, then element i of B is matched to element j of A .

Under most common optimality criteria, an optimal P cannot be found in polynomial time. Several authors have obtained approximations to P by finding a matrix \tilde{P} which has certain structural properties and minimizes the following objective for some matrix norm u :

$$\| \mathcal{A} - \tilde{P}^\top \mathcal{B} \tilde{P} \|_u$$

In [12], Umeyama solves this problem for \tilde{P} orthogonal (and thus square) for $u = 2$. Almohamad and Duffuaa present a linear programming formulation in [13] that solves this problem for \tilde{P} doubly-stochastic and $u = 1$. In order to obtain a permutation matrix P from \tilde{P} , both authors apply the Hungarian algorithm to \tilde{P} . The *Hungarian algorithm*, originally formulated by Kuhn in 1955, computes a maximum weight matching between two sets, each with n elements, in $\mathcal{O}(n^3)$ time [14].



Figure 1: The prototype hub-authority graph G_{HA} .

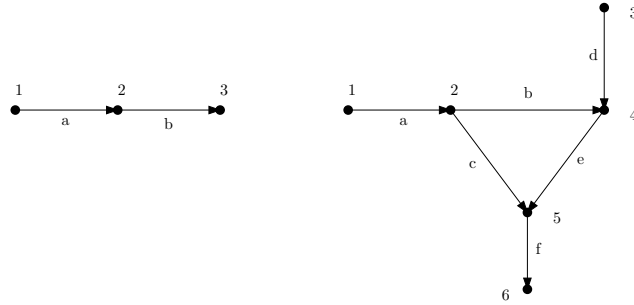


Figure 2: Two directed graphs.

| X | | | | Y | | |
|-------|-------|-------|-------|-------|-------|-------|
| nodes | 1 | 2 | 3 | edges | a | b |
| 1 | 0.124 | 0 | 0 | a | 0.265 | 0 |
| 2 | 0.348 | 0.445 | 0 | b | 0.426 | 0.297 |
| 3 | 0.157 | 0.054 | 0 | c | 0.320 | 0.389 |
| 4 | 0.094 | 0.563 | 0.193 | d | 0.336 | 0.115 |
| 5 | 0 | 0.338 | 0.340 | e | 0.202 | 0.445 |
| 6 | 0 | 0 | 0.094 | f | 0 | 0.202 |

Table 1: X , the node similarity scores, and Y , the edge similarity scores for the graphs of Figure 2. As discussed in Section 2, higher scores denote greater similarity.

The node scoring method presented in the previous section yields a matrix of similarity scores between nodes taken pairwise from each graph, which naturally suggests matching the nodes of two graphs by computing a maximum weight matching on this matrix of scores.

We shall consider the following illustrative problem: identify the location of a subgraph within a larger graph. To explore this problem empirically, define a graph G_A by generating an $n \times n$ node-node adjacency matrix \mathcal{A} ; each entry in \mathcal{A} is set to 1 with a specified probability p . This is the definition of an Erdős-Rényi $G_{n,p}$ random graph [15]. A set of $m \leq n$ vertices of G_A is selected; the subgraph induced by these m nodes is denoted G_B . G_B may or may not be connected. Next, the node similarity matrix between G_A and G_B is computed, using the scoring method described in Section 2. Finally, the Hungarian algorithm is applied to the node similarity matrix to obtain a matching between the nodes of the subgraph and a subset of the nodes of the original graph that maximizes the sum of matched scores.

The results presented here were generated in MATLAB®, and utilized Borlin’s implementation of the Hungarian algorithm [16]. Observe that, since $m \leq n$, the application of the Hungarian algorithm requires padding the matrix of node similarity scores with extra entries; these fictitious scores are set to a negative number. A success is counted for every match between a subgraph node and its *original* identification in the larger graph; note that this kind of accounting yields a lower bound on the success of the matching, since better or equally good matches that do not correspond to the original identification are marked as failures.

As a first test of this concept, this procedure has been applied to 15-node graphs with variably-sized subgraphs and variable edge probability (connectivity) parameter p . For each value of p , 500 subgraphs of the specified sizes were generated and the average proportion of successful matches was recorded. The results are given in Figure 3.

Figure 3 also depicts the expected proportion of correct matches if the subgraph nodes were *randomly assigned* to nodes in the original graph. The computation of this lower bound is similar in concept to the *matching hats problem*, in which n party guests leave their hats in a room; after the party, the hats are randomly redistributed. Now, imagine that $n - m$ of the hats are stolen, leaving m remaining. The hats are then distributed randomly to the guests (analogously, in the subgraph matching problem, subgraph nodes are matched to nodes in the larger graph). One can show that

$$E[\# \text{ of correct matches}] = \frac{m}{n}. \quad (7)$$

Thus, the expected *proportion* of correct matches for a subgraph of size m is $1/n$, the constant function of subgraph size plotted in Figure 3. The details of this computation can be found in [17], which also contains the performance bounds for the matching scenarios described in the following two subsections. It is encouraging that the similarity-based matching performance is a substantial improvement over the expected performance of random matching.

Some interesting features of the data of Figure 3 can be observed. First, it appears that higher initial connectivities lead to poorer performance. Additionally, better performance is achieved with larger subgraphs. Finally, note that when the subgraph is the *entire* graph, the procedure performs the matching correctly in all of the trials.

Nodes with type labels

Often in graph matching applications, there exists side information about the attributes of nodes and edges. For example, the proteins that constitute the nodes of a protein interaction network might be classified by their type. When attempting to align these networks across species, one might want to only match proteins that share the same type. The node matching procedures that have been discussed in the preceding section have used only graph topology to identify similarities; including node identity information to penalize or reward certain matches may improve performance.

We augment our subgraph matching procedure by assigning each node in the graph G_A one of s labels with equal probability. A random subgraph G_B is identified, retaining the node labels from G_A . Once the node similarity matrix between G_A and G_B has been computed, scores between nodes with different labels are assigned a large negative weight; the maximum weight matching is then computed.

Performance results for this approach are presented in Figure 4 as histograms of the number of correct node matches out of 500 trials for subgraphs of different sizes. The averages for each number of labels and each subgraph size are drawn as vertical lines. The expected performance results under the related random matching scenario, as derived in [17], are represented by the dashed vertical lines. Clearly, increasing the number of labels improves performance.

We might also ask what will happen if there are only two node labels, which are distributed with asymmetric probabilities. Figure 5 depicts histograms of subgraph matching performance for a graph with two node labels, $L \in \{0, 1\}$, which are distributed with the probabilities indicated. As expected, the more evenly the two labels are distributed, the more node identity information is communicated by the label.

Nodes with unique labels

Often, nodes will have unique identifiers; for example, there may exist reference nodes that can be uniquely identified in pattern recognition problems. To model this scenario, a random graph G_A with n nodes is generated, then s of its nodes are labeled with unique identifiers. The remaining $n - s$ are given the same label. A random subgraph G_B is removed, carrying the node labels along. In general, the subgraph G_B will contain fewer than s of the uniquely labeled nodes. The node similarity matrix between G_A and G_B is computed, then edited by setting the similarity scores between nodes with mismatched labels to a large negative number. The procedure continues as described in Section 3.

In Figure 6, each curve represents the performance for a particular value of s as it is varied from 0 to 14 on a 15 node graph. The dashed horizontal lines represent the expected performance under the related random matching scenario as s is varied from 0 to 14. Matching performance improves steadily as the number of unique labels is increased.

4 Conclusions

Exploring the information about the graphs G_A and G_B contained in the similarity matrices is an ongoing task. What kinds of structural features common to G_A and G_B can be inferred from patterns in the similarity scores?

The node matching procedure can be applied to pairs of isomorphic graphs to generate self-similarity matrices; in exploratory tests, an isomorphic mapping was always among the maximum weight matches. The self-similarity matrices of some graphs communicate very little or no interesting information; for example, any pair of undirected bipartite graphs (including those that represent the point-line incidence graphs associated with projective planes) has a set of node scores that are the same between every node pair. While projections onto the dominant eigenspace associated with the matrix M as defined in Eq. 5 do not reveal any interesting structural information, it is possible that the subspaces associated with the strictly smaller eigenvalues of M do encode such information. For many kinds of graphs, the spectrum of the node-node adjacency matrix is well-characterized; using the machinery of spectral graph theory to explore the matrix M would be worthwhile.

Novel applications might arise from any natural phenomenon with a graph structure: recall that Kleinberg’s motivation in [6] was an improved web search tool, while Blondel et. al [11] applied their method to automatic synonym extraction from a dictionary graph. Another intriguing application of graph similarity methods is the identification of functionally similar subgraphs within maps of protein interactions compared across different species. We have explored the identification of such subgraphs in [18], and Holme and Huss have used a vertex similarity scoring method to predict the function of proteins in *S. cerevisiae* in [19]. Finding new applications for this kind of similarity scoring might inspire some interesting new heuristics, or an entirely new approach.

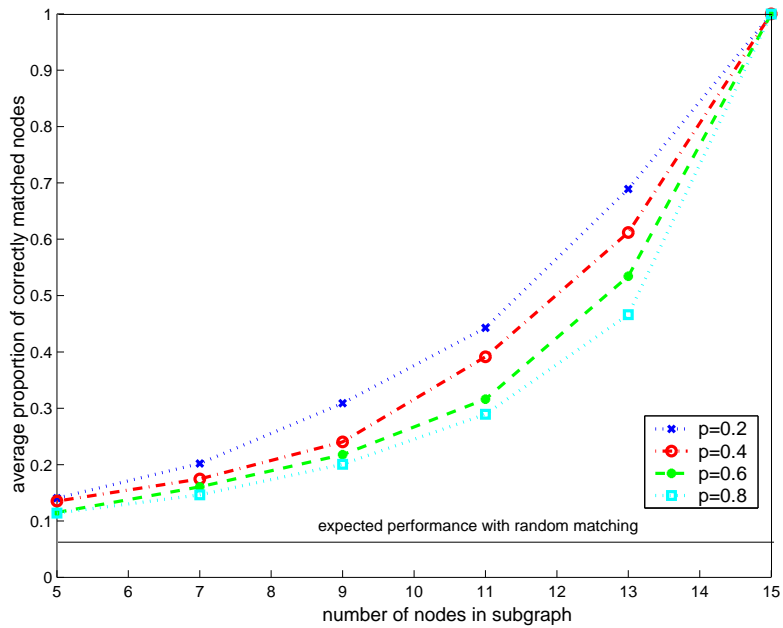


Figure 3: Subgraph matching performance on a 15-node Erdős-Rényi random graph with edge probabilities $p = 0.2, 0.4, 0.6, 0.8$, using the Hungarian method applied to the node scores obtained from Eqs. 1 and 2, and averaging over 500 trials.

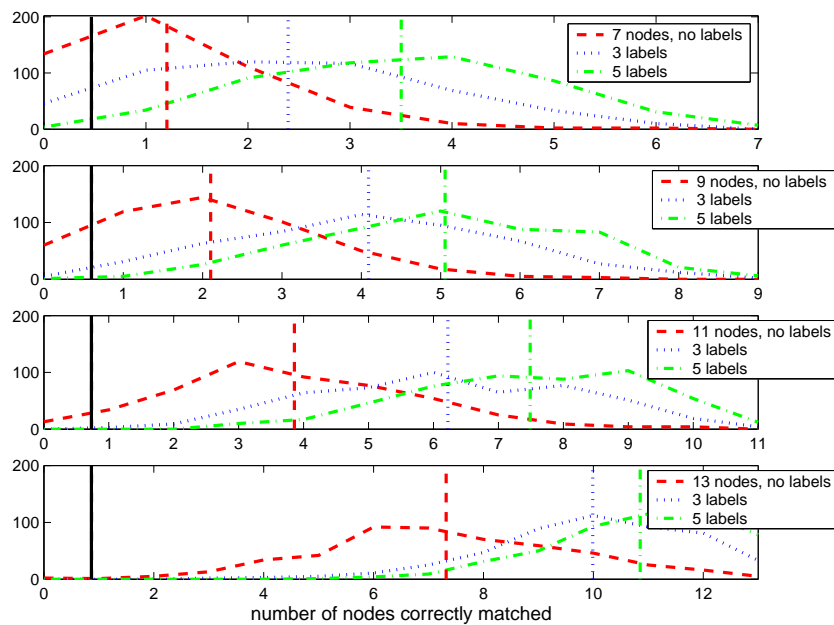


Figure 4: Matching performance using 0-5 node labels on a 15-node Erdős-Rényi random graph with connectivity 0.5, with subgraphs of size 7, 9, 11 and 13 nodes. The mean of each histogram is represented by a vertical line, while the expected performance under the analogous random matching scenario is given by the solid vertical line.

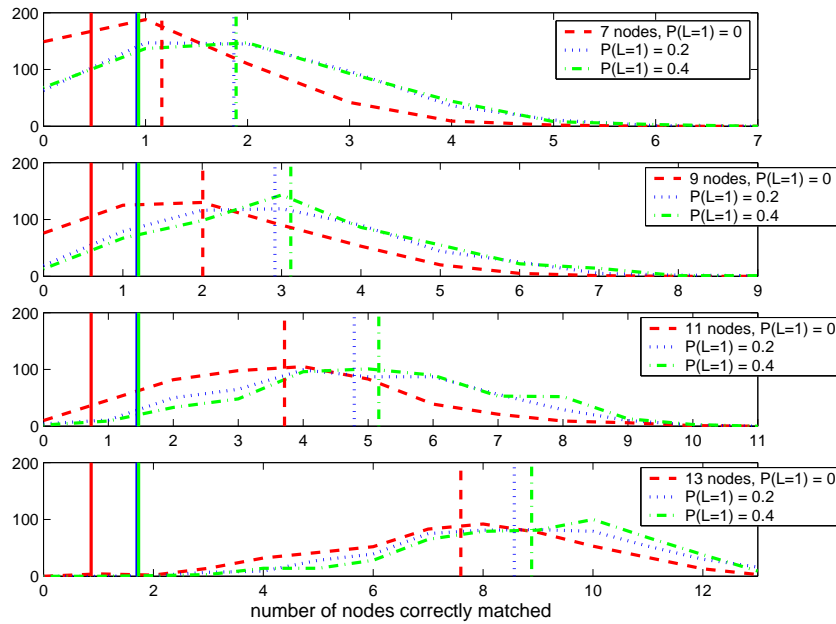


Figure 5: Matching performance using 2 node labels distributed with different probabilities on a 15-node Erdős-Rényi random graph with connectivity 0.5, with subgraphs of size 7, 9, 11 and 13 nodes. The mean of each histogram is represented by a solid line, while the expected performance under the analogous random matching scenario is given by the solid vertical lines.

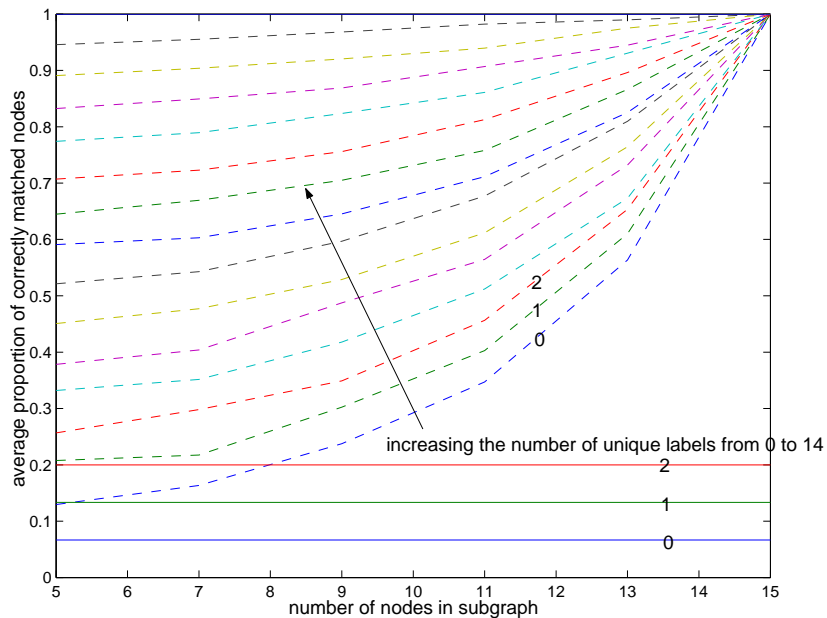


Figure 6: Matching performance using varying numbers of unique labels. The dashed lines represent the expected performance of a random matching scenario for s from 0 to 14.

Acknowledgment

This material is based upon work supported in part by a National Science Foundation Graduate Research Fellowship. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation. Additional support for this work was provided by the DoD AFOSR URI for ‘Architectures for Secure and Robust Distributed Infrastructures,’ F49620-01-1-0365 (led by Stanford University). We thank Vincent Blondel and Paul Van Dooren for their helpful comments and discussion.

References

- [1] Ullman, J.R. (1976). An algorithm for subgraph isomorphism. *J. Assoc. Computing Machinery*, v. 23(1), 31-42.
- [2] Pelillo, M. (1999). Replicator equations, maximum cliques and graph isomorphism. *Neural Computation*, v. 11(8), 1933-1955.
- [3] Bunke, H. (1999). Error correcting graph matching: on the influence of the underlying cost function. *IEEE Trans. Pattern Analysis and Machine Intelligence*, v. 21, 917-922.
- [4] Fernandez, M.L., Valiente, G. (2001). A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, v. 22, 753-758.
- [5] Pelillo, M. (2002). Matching free trees, maximal cliques and monotone game dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, 1535-1541.
- [6] Kleinberg, J.M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, v. 46, 614-632.
- [7] Melnik, S., Garcia-Molina, H., Rahm, A. (2002). Similarity flooding: a versatile graph matching algorithm and its application to schema matching. *Proceedings of the 18th International Conference on Data Engineering*, San Jose.
- [8] Jeh, G., Widom, J. (2002). SimRank: A measure of structural-context similarity. *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining*, Edmonton.
- [9] Heymans, M., Singh, A. (2003). Deriving phylogenetic trees from the similarity analysis of metabolic pathways. *Bioinformatics*, v. 19(Supp.1), 138-146.
- [10] Leicht, E., Holme, P., Newman, M. (2006). Vertex similarity in networks. *Physical Review E*, in press.
- [11] Blondel, V., Gajardo, A., Heymans, M., Senellart, P., Van Dooren, P. (2004). A measure of similarity between graph vertices: applications to synonym extraction and web searching. *SIAM Review*, v. 46(4), 647-666.
- [12] Umeyama, S. (1988). An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 10(5), 695-703.
- [13] Almohamad, H., Duffuaa, S. (1993). A linear programming approach for the weighted graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 15, 522-525.
- [14] Kuhn, H. (1955). The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, v. 2, 83-97.
- [15] Erdős, P., Rényi, A. (1959). On random graphs. *Publicationes Mathematicae*, v. 6, 290-297.

- [16] Borlin, N. assignprob.zip. Available from MATLAB®Central File Exchange. <http://www.mathworks.com/matlabcentral/fileexchange/>. Last accessed: 12/22/2005.
- [17] Zager, L., Verghese, G. (2006). Caps and robbers - what can you expect? *College Mathematics Journal*, to appear.
- [18] Zager, L. (2005). *Graph similarity and matching*. Masters' thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.
- [19] Holme, P., Huss, M. (2005). *Role-similarity based functional prediction in networked systems: application to the yeast proteome*. *Journal of the Royal Society Interface*, v. 2(4), 327-333.