Graph Similarity and Matching

by

Laura Zager

B.A., Mathematics (2003)
B.S., Engineering (2003)
Swarthmore College

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

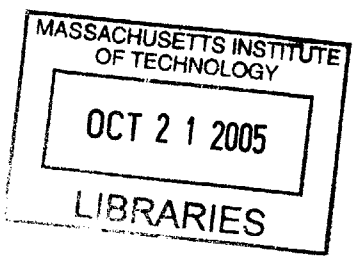
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2005

Author_____
Department of Electrical Engineering and Computer Science
May 19, 2005

Certified by_____
George Verghese
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by_____
Arthur C. Smith
Chairman, Department Committee on Graduate Students



Graph Similarity and Matching

by

Laura Zager

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2005, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Measures of graph similarity have a broad array of applications, including comparing chemical structures, navigating complex networks like the World Wide Web, and more recently, analyzing different kinds of biological data. This thesis surveys several different notions of similarity, then focuses on an interesting class of iterative algorithms that use the structural similarity of local neighborhoods to derive pairwise similarity scores between graph elements. We have developed a new similarity measure that uses a linear update to generate both node and edge similarity scores and has desirable convergence properties.

This thesis also explores the application of our similarity measure to graph matching. We attempt to correctly position a subgraph G_B within a graph G_A using a maximum weight matching algorithm applied to the similarity scores between G_A and G_B. Significant performance improvements are observed when the topological information provided by the similarity measure is combined with additional information about the attributes of the graph elements and their local neighborhoods. Matching results are presented for subgraph matching within randomly-generated graphs; an appendix briefly discusses matching applications in the yeast interactome, a graph representing protein-protein interactions within yeast.

Thesis Supervisor: George Verghese
Title: Professor of Electrical Engineering and Computer Science

Acknowledgements

I'd like to express my gratitude to my advisor, George Verghese, for his guidance in my early exploration and in the development of this thesis. Special thanks to Paul Van Dooren of the Department of Mathematical Engineering at the Université catholique de Louvain for his HPCES seminar, which inspired our work, and for his subsequent correspondence. Maya Said's suggestions were immensely helpful and let us dabble in biological applications without getting in over our heads. Thanks also to Sandip Roy and Pablo Parrilo for their fresh ideas and perspectives.

My family is amazing - Mom, Dad, and Lisa, plus Gramma, Mike, Cheri, Larry, Robin, Melissa and Karen. They are hilarious and supportive and wonderful beyond words. I don't think I would have acclimated to MIT nearly as quickly without the help of Al Oppenheim and EGCS-I, nor without Tushar and everyone in LEES. And many thanks to my great friends, both here at MIT and from Swarthmore and beyond: Shubham, Vasanth, Holly, Akshay, Keith, Zach, Jamie, Andy, Alyssa, Aimee, Christine and Katie.

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 author and do not necessarily reflect the views of the National Science Foundation. Additional support for this work was provided by the Department of Defense University Research Initiative on 'Architectures for Secure and Robust Distributed Infrastructures'.

Contents

Chapter 1

Introduction

GRAPH THEORY, the study of binary relationships among elements of a set, has undergone tremendous changes since its formal beginnings in the 18th century. Early work, beginning with Euler's solution to the Kőnigsberg bridge problem, was largely focused on proving combinatoric and existence statements about graph structures. A major advance occurred in 1959 when a paper by Erdős and Rényi introduced the *random graph*, whose nodes are connected to each other according to a probabilistic model. Throughout these periods, graph theorists focused principally on graphs that were theoretically constructed, and sought to answer questions about their properties.

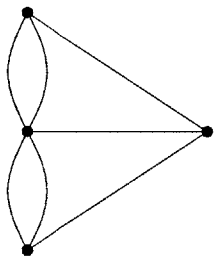


Figure 1.1: *The Kőnigsberg bridge problem: Euler was the first to prove the impossibility of starting from any node in this graph, passing through every edge exactly once, and returning to the starting point.*

Over the past twenty years, however, researchers have revived graph theory to answer questions about real graphs that arise in the natural world. This revival has come in response to several related developments. The first is the *data explosion* that has taken place in many fields of research, fueled by the continual improvement in computing and data storage. Particularly in biology, where automatic sequencing and microarray technology have revolutionized experimental work, researchers have taken full advantage of these improvements, generating data quickly and on a massive scale. Often, these data can be imbued with a graph structure that reveals interesting relationships and suggests further lines of inquiry. Scientists faced with these kinds of data

graphs require algorithms that can search through large datasets quickly and return the relevant information.

A second driving force behind graph theory's recent re-emergence is increased *collaboration* between researchers in previously disparate fields, largely due to the ease of electronically disseminating information. This new collaboration has revealed that the graph structures seen by biologists, physicists and sociologists often have the same interesting and peculiar features. Discoveries of this kind certainly open the lines of communication between different academic communities, but perhaps more importantly, also compel researchers to return to the pure theory to search for fundamental explanations.

A quick and readable survey of modern attitudes toward graph theory can be found in [H00.1] and [H00.2]. The following section is devoted to a brief discussion of many of the interesting graph theory applications that have arisen across a broad array of scientific fields. This survey will motivate an introductory discussion of the notion of the *similarity* of two graphs, the exploration of which is to be the focus of this paper.

1.1 Examples and applications

To begin to understand which graph theoretic concepts are useful, it is worthwhile to sample some uses of graph theory in practical research. This section will do its best to avoid technical notation and results, but definitions of any concepts that may arise can be found in Chapter 2.

1.1.1 Chemistry

The application of graph theory to problems in chemistry has a long history. Chemical compounds have a natural graph structure, with nodes representing different types of atoms and edges representing inter-atomic bonds. One early survey of the use of graph theory in chemistry is given by Balaban in [B85], in which the author emphasizes the essentially graph theoretical nature of many contemporary problems, including identifying isomers, planarity of molecules, and reaction graphs. Today, the question facing computational chemists is not whether a graph theoretic approach is appropriate, but which graph-based algorithm should be selected.

When experimental chemists develop a new compound, their first step is to compare its molecular struc-

ture to a database of compounds and look for those which are structurally identical, or have structurally identical sub-components. In graph theory, this is referred to as the *maximum common subgraph* problem, and is discussed in greater detail in Section 3.3. A recent survey of the state of the art in solving maximum common subgraph problems for chemical compounds is given by Raymond and Willett in [RW02]. A related approach is taken by Kanehisa et al. in [HOGK03] with their development of a similarity measure for chemical structures that uses structural subgraph matching to identify biochemically meaningful features in a database of metabolic compounds.

1.1.2 Biology

Biological systems are complex networks with many different kinds of interacting units, from the most 'macro' population level to the most 'micro' molecular scale. Detangling this web of interactions has been made more plausible over the last twenty years with the development of many breakthrough technologies that have allowed researchers to probe the activating and inhibiting relationships between biological components. In many ways, the forefront of biology is no longer the collection of experimental observations, but the intelligent analysis of existing data. In [KB03], Kanehisa and Bork reflect on the challenges facing biologists as they develop models that have a sound underlying structure and predictive value. Mapping and understanding the nature of biological networks is central to addressing these challenges.

Different kinds of graphs constructed from biological data require different types of analyses. One example of biological data with a graph structure is a mapping of metabolic networks into *enzyme graphs*: biological enzymes serve as nodes in the graph, with an edge existing from enzyme e_1 to enzyme e_2 if e_1 catalyzes a reaction whose product is the substrate for e_2. In Section 4.3.3, we will have much more to say about one particular use of these graphs in our discussion of the work of Heymans and Singh in [HS03]. More generally, the *protein-protein interactions* of any species can be represented as a graph; one such example is given in Figure 1.2.

Another graphical representation of a metabolic network is one that represents the differential equations governing some portion of the metabolic pathway of an organism. In these graphs, each node represents a molecule involved in a particular metabolic process, which is subject to mass conservation and thermody-

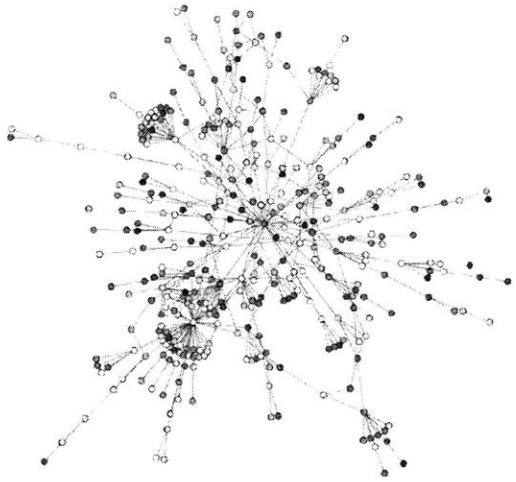


Figure 1.2: *A graphical representation of a protein-protein interaction network (provided by [V03]).*

namic requirements represented by the edges. In these kinds of networks, one is often interested in exploring different paths through the network in order to predict behavior or explain experimental results. In each of [SLP00] and [PRPF03], the authors present different methods for finding representations of these metabolic networks in terms of essential pathways within the network, then use these essential pathways to provide some biological insight.

Biologists are also interested in a more 'global' or 'aggregate' view of graphs of biological data. In [CLDG03], Chung et al. develop a probabilistic model for gene duplication that attempts to explain the empirically-observed degree distribution (see Chapter 2) of many biological networks.

1.1.3 Computer vision

Computer science has long used graph theory to represent network and data structures, but one of the most interesting applications of graph theory is to pattern recognition and computer vision. Given two images, image processing software can often identify interesting features, and then describe the relationship among these features, for instance, in terms of distance and angle. An example of this kind of processing is given in Figure 1.3. This naturally leads to the representation of the image as an *attributed graph.*

An excellent and thorough overview of graph theory in pattern recognition is given by Conte et al. in

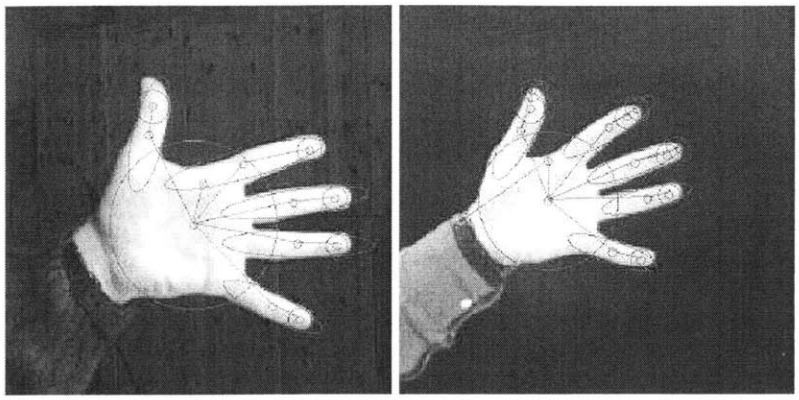


Figure 1.3: *Two related images, with indicated features (provided by [DSDK04]).*

[CFSV04]. In addition to discussing graph matching techniques, the authors describe a number of applications that rely heavily on graph representations, including optical character recognition and biometric identification. Many of these problems involve trying to find a subject graph within a database of graphs, the same task that arises with chemical compounds in databases.

1.1.4 Social networks

Much has been made in the past few years of the *small world phenomenon*, which occurs when a large, sparsely connected network has a surprisingly low average distance between any two randomly chosen nodes. This phenomenon was first observed and studied in social networks by the psychologist Stanley Milgram in the 1960s [TM69]. Milgram (most famous for his disturbing experiments investigating obedience to authority) devised a series of experiments to quantify the average number of acquaintances between any two people in America. In his experiments, a randomly selected person A in Nebraska was asked to route a letter to a randomly selected person B in Boston by passing the letter to an acquaintance who might know better how to reach B. The letters traveled along the acquaintance network, many of them reaching the target person. Among letters that successfully reached the target, Milgram found that the average number of people that it had to pass through was 5.7, an intriguingly small number for strangers situated 1300 miles apart.

The network theory needed for a more rigorous analysis of these kinds of results did not come together until the late 1990s, with work by Duncan Watts. In [W99], Watts presents several theoretical models for

networks, each with different rules for how nodes may be connected, and examines these models for evidence of the small world phenomenon. Watts' work has inspired researchers in many fields to search for this phenomenon in their own data and examine its implications. For example, Davis, Yoo and Baker in [DYB03] perform an analysis of the relationships between board members of the top American corporations and find evidence of a small-world network. Most interestingly, the small world nature of these networks has persisted over the last twenty years, amidst considerable change in the selection procedure for board members and the duties required. The implication of this phenomenon is that new ideas in corporate strategy can move quickly through the corporate 'elite,' much like an infection, via person-to-person contact.

Certainly, the application of graph theory to social networks is not limited to observing occurrences of the small world phenomenon. Another compelling concept is that of *viral marketing*, in which a group desires to influence an entire population by selectively influencing well-connected members of the population. The questions of how best to define a network that models a target population, and from there to select the most influential members and estimate the network's behavior, are of much interest. For one set of models and performance results, see Kempe, Kleinberg and Tardos' work [KKT03].

1.1.5 World Wide Web

The Web is a fascinating graph, constantly undergoing editing by millions of independent users across the globe. Edges between nodes in the Web graph, representing links between webpages, can represent many different kinds of relationships, including the personal, professional, and navigational. The Web often serves as a proxy measure of the social structure of a community; for example, the graph of homepage connections between MIT students represented in Figure 1.4. The Web provides a dual set of challenges for researchers; to understand the statistics and dynamics of Web growth and connectivity, and to use this knowledge to improve tools for information retrieval from the Web.

One step in this direction is provided by Dill et al. in [DKMR02], who performed extensive sampling of the Web and found evidence of a fundamentally *fractal* organization; the same types of structures (and hence the same statistics) can be seen at varying scales of the Web graph. Their contention is that this repeated structure should simplify the design of web-based algorithms and improve search quality.

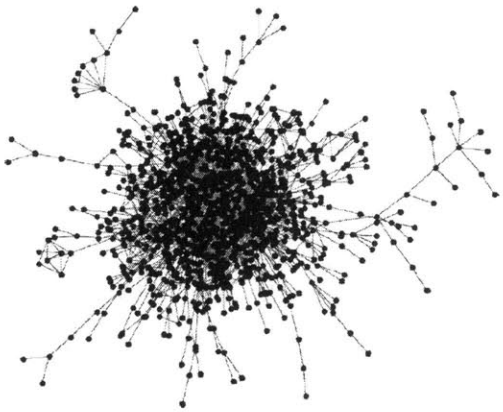


Figure 1.4: *The network of connections among MIT student homepages (taken from [AA03]).*

The sheer size of the Web graph is astounding: one lower bound is 8 billion documents in November, 2004, given by the number of sites in the Google search engine database [G04]. Google's search algorithm, the most commonly used in the world, was originally proposed by Brin and Page in 1998 [BP98]. The power of their approach derives from not only considering the *text* in a Web document and its relevance to a given query, but how the *link structure* of the graph as a whole differentiates between important and marginal sources of information. Another web search tool that relies heavily on link structure is the HITS algorithm proposed by Kleinberg [K99], which we will discuss in detail in Chapter 4.

1.2 Similarity

One common thread that runs though many of the examples in the previous section is the need to be able to compare two graphs and somehow assess their *similarity*. Certainly, what one means by 'similarity' is particular to the application; there are many kinds of similarities one may be interested in between two graphs. A first list of such similarity questions might include:

- Are these graphs identical copies of each other?
- How many changes must I make to one graph to change it into the other?
- Do each of these graphs contain an identical subgraph? How large is this subgraph?

- Can we contain both of these graph in a single larger graph? How small can this graph be?
- If we allow the graph to change over time, can we assess whether two graphs were generated from the same set of evolutionary rules from a common ancestor graph?
- How much does the neighborhood of a given node in one graph look like the neighborhood of a given node in another graph? Can we compare the neighborhoods of edges?

Each of these questions corresponds to a distinct similarity notion. Indeed, having many different notions of similarity available allows a researcher to select the most appropriate measure for his or her application. There exists an array of answers for each of these questions, many of which will be discussed in Chapter 3.

Chapter 2

Graph terminology

THIS CHAPTER will outline some of the important definitions and concepts of graph theory that we will utilize throughout the thesis. The final section contains a brief review of some standard linear algebra results that will be useful.

2.1 Graph fundamentals

A *graph* $G(V, E)$ consists of a set of *vertices* (interchangeably called *nodes*), V, and a set of *edges*, $E \subseteq V \times V$. A graph may be *directed* or *undirected*; in an undirected graph, for $u, v \in V$, then $(u, v) \in E$ implies that $(v, u) \in E$. Two pictorial examples of graphs are shown below in Figures 2.1 and 2.2.

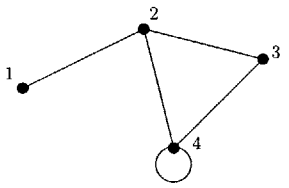


Figure 2.1: *An undirected graph, G_u.*

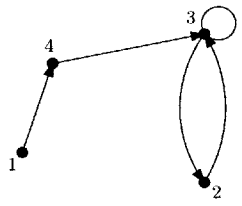


Figure 2.2: *A directed graph, G_d.*

For an edge (u, v), the *source* of the edge is node u and the *terminus* of the edge is node v. In an undirected graph, both nodes u and v are source and terminal nodes.

A *path* through a graph is a sequence of nodes (with no repetitions) connected by edges. For example, $2 \rightarrow 3 \rightarrow 4$ is an admissible path in the graph of Figure 2.1, but not for the graph of Figure 2.2 because it does not respect edge orientation. A *cycle* is a path that ends on its starting node. A connected graph (see Section 2.6 for a definition) with no cycles is called a *tree*.

2.2 Node-node adjacency matrices

A convenient way to represent a graph is a *node-node adjacency matrix* (also referred to simply as an *adjacency matrix*). Consider a graph, $G_A = G(V_A, E_A)$. If the cardinality of V_A is n_A, then the adjacency matrix $\mathcal{A}$ of this graph is an $n_A \times n_A$ matrix in which entry $[\mathcal{A}]_{ij}$ is equal to 1 if and only if $(i, j) \in E_A$, and is equal to 0 otherwise. Observe that the adjacency matrix of an undirected graph will always be symmetric. The adjacency matrices of the graphs in Figures 2.1 and 2.2 are given below.

$$\mathcal{A}_u = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Table 2.1: *Adjacency matrix of Figure 2.1.*

$$\mathcal{A}_d = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Table 2.2: *Adjacency matrix of Figure 2.2.*

Every node i in a graph has an *out-degree* and an *in-degree*, the former being the number of edges that begin at i and the latter the number of edges that terminate at i. These values can be found by simply summing over the rows and columns of the node-node adjacency matrix, respectively. Note that in an undirected graph, the out-degree and in-degree of a node are equal. One measure of a graph's global structure is the *degree distribution*, a histogram of the degrees of each of its nodes.

2.3 Node-edge adjacency matrices

Another common matrix representation of a graph is the *node-edge adjacency matrix*. For a graph $G_A = G(V_A, E_A)$, denote its node-edge adjacency matrix by A, a $|V_A| \times |E_A|$ matrix, with each row of A corresponding to an element of V_A and each column corresponding to an element of E_A. First, let $s_A(i)$ denote the source of edge i, and let $t_A(i)$ denote the terminus of edge i. Then we can define the node-edge adjacency matrix A as follows:

$$[A]_{ij} = \begin{cases} -1 & s_A(j) = i \\ 1 & t_A(j) = i \\ 0 & \text{else} \end{cases}$$

Observe that this definition is not well-defined for a *self-loop* in the graph G_A, i.e. an edge (i, i). Thus, we

will introduce a further refinement of the node-edge adjacency matrix that resolves this issue: the representation of a graph G_A by a pair of matrices, the *edge-source matrix* A_S and the *edge-terminus matrix* A_T. These matrices can be defined as follows:

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

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

For graphs without self-loops, $A = A_T - A_S$. This kind of representation has some useful and interesting properties, including the following:

- $\mathcal{A} = A_S A_T^\top$. To see this, observe that entry $[A_S A_T^\top]_{ij}$ is the inner product of the ith row of A_S and the jth row of A_T, which counts the number of edges from node i to node j.

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

- $D_{A_T} \equiv A_T A_T^\top$ is diagonal with the *in-degree* of each node in the corresponding diagonal entry.

- $A_S^\top A_T$ is a particular kind of *edge-edge adjacency matrix* which tell us when one edge originates at another's terminal node. Specifically,

$$[A_S^\top A_T]_{ij} = \begin{cases} 1 & s_A(i) = t_A(j) \\ 0 & \text{else} \end{cases}$$

  This matrix is symmetric, with a '1' in the ith diagonal entry if edge i is a self-loop.

- The matrices $A_S^\top A_S$ and $A_T^\top A_T$ are also kinds of edge-edge adjacency matrices, which tell us when two edges start or terminate at the same node, respectively. Explicitly, these matrices are:

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

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

These matrices always have '1' entries on the diagonal, and are symmetric.

The representation of a graph by two node-edge matrices will be the most computationally convenient in our later work on iterative similarity methods.

2.4 Weighted graphs and attributed graphs

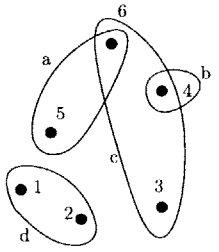
In a graph representing a social network, for example, connections between individuals are rarely best described as '1's and '0's; it is useful to be able to distinguish between a strong friendship and a weak acquaintanceship. This naturally leads to a notion of *weighted edges*, in which each edge i of a graph has a weight w_i (often restricted to the interval $[0, 1]$). We can generalize the node-node adjacency matrix by allowing the ijth entry to take the value of the weight of edge (i, j). Weighted graphs also naturally arise in the representation of electrical circuits, with the weight of an edge representing a resistance value.

It is also possible to associates weights to the *nodes* in a graph: one could imagine a scenario in which the weight of a node indicated that node's relative importance to the graph, perhaps in modeling the hierarchy in a corporation. One way of incorporating these weights into a graph model is to left-multiply the node-node adjacency matrix by a diagonal node weight matrix, but there are certainly others; the best choice of a node weighting method depends upon the application.

A higher-level approach to weighting nodes and edges in graphs is to label each node and/or edge with a vector of *attributes*. Attributes could be drawn from any class of descriptors; these could simply be weights as discussed previously, or they could be text strings or even functions. Attributed graphs naturally arise in image processing applications, where nodes might be associated with a point in an image and be attributed with an RGB color vector as an attribute, while edges may have distance and angle attributes.

2.5 Hypergraphs

One natural extension of the graphs that we've discussed, in which edges are *pairs* of nodes, is a *hypergraph*, in which edges are *subsets* of nodes. In a hypergraph, two nodes are adjacent if there exists a hyperedge containing both nodes. The most appropriate representation of a hypergraph is a node-edge adjacency matrix with multiple 1's allowed in each column.[1] An example of a hypergraph is depicted in Figure 2.3, with nodes labeled by numbers and hyperedges labeled by letters, alongside its node-edge adjacency matrix, H.



$$H = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Figure 2.3: *A hypergraph G_H and its node-edge adjacency matrix H.*

2.6 Statistics of graphs

For simplicity, we will state the following properties with respect to an *undirected graph*, bearing in mind that there exist generalizations of each to the directed case. For additional interesting graph properties, see Watts [W99] and Buckley and Harary [BH90].

A graph is *connected* if there exists a path between any pair of nodes. The nodes of any graph can be partitioned into *connected components*, each of which is connected. An example of an undirected graph with two connected components is given in Figure 2.4. A *geodesic* between nodes s and t is a path of shortest length from s to t. Note that a geodesic between two nodes is not necessarily unique. The *diameter* of a connected graph is the length of its longest geodesic.

In a communication network, we might be interested in identifying which nodes are critical to the network's operation, in the sense that removal of one of these nodes jeopardizes the connectivity of the network. One measure of the criticality of node i is its *betweenness centrality*, $C(i)$. Define σ_{st} to be the number of

[1] Berge's canonical text on hypergraphs [B73] actually defines the adjacency matrix as the transpose of this definition; we've chosen this representation for continuity with the standard graph case.

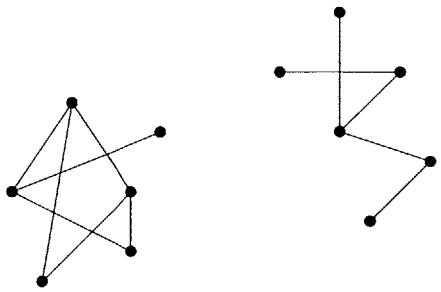


Figure 2.4: *An undirected graph with two connected components.*

geodesics from node s to node t, and $\sigma_{st}(i)$ to be the number of geodesics that go through node i. Then we can define $C(i)$ as follows:

$$C(i) = \sum_{s \neq t} \frac{\sigma_{st}(i)}{\sigma_{st}}$$

If $C(i) = 1$, removal of node i is guaranteed to disconnect the graph.

Another node statistic is the *clustering coefficient* of a given node, which measures how connected a node's neighbors are to each other. This statistic might be of interest in an friendship network, where the clustering coefficient provides a measure of how tightly knit one's social group is. Define the *neighborhood* of node i as the set of nodes j such that edge (i, j) exists, and denote the number of neighbors of node i by k_i. Finally, denote the number of connections between these neighbors by $E(i)$. Observe that the maximum number of possible edges between the neighbors of i is given by $\binom{k_i}{2}$. Then we can define the clustering coefficient of node i, $\gamma(i)$, as

$$\gamma(i) = \frac{E(i)}{\binom{k_i}{2}}$$

In a *complete* graph, one in which every distinct pair of nodes is connected by an edge, the clustering coefficient of every node is 1.

2.7 Linear algebra review

Before we begin a technical discussion, it is worth reviewing some useful linear algebra results and standardizing our notation.

Recall the following definitions:

- A matrix M is *symmetric* if $M = M^\top$, where $^\top$ denotes matrix transposition.

- The *trace* of a matrix M, denoted by tr(M), is the sum of the diagonal elements of M.

- A matrix M is *positive definite* if $x^\top Mx > 0$ for every vector $x \neq 0$. M is *positive semi-definite* if $x^\top Mx \geq 0$ for every $x \neq 0$.

- The *Perron root* of a square, component-wise non-negative matrix M, denoted by $\rho(M)$, is the real eigenvalue of M with largest magnitude.

- The *p-norm* of a vector $x \in \Re^n$, denoted by $\| x \|_p$, is defined as

$$\| x \|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

- The *Frobenius norm* of an $n \times m$ matrix M, denoted by $\| M \|_F$, is defined as

$$\| M \|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{m} [M]_{ij}^2 \right)^{1/2}$$

- For two vectors $v, w \in \Re^n$, their *Euclidean inner product*, $< v, w >$, is defined as $v^\top w$.

- Let S be a subspace of $\Re^n$. The *orthogonal projection onto* S of a vector $x \in \Re^n$ is the unique element x^* in S that has the smallest Euclidean distance (2-norm) to x. Given x, we can find x^* via the *orthogonal projector*, Π. To construct Π, let $v_1, v_2, .., v_m$ form an orthonormal basis for S. Then $x^* = \Pi x$ where $\Pi = PP^\top$ and $P = \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_m \\ | & | & & | \end{bmatrix}$.

- The *invariant subspace* of a symmetric matrix $M \in \Re^{n \times n}$ associated with the eigenvalue λ is the span of all $v \in \Re^n$ such that $Mv = \lambda v$.

- If $A = \{a_{ij}\}$ is an $n \times n$ matrix, and $B = \{b_{ij}\}$ is an $m \times m$ matrix, then the *Kronecker product*, $A \otimes B$ is the $nm \times nm$ matrix defined by:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}B & a_{n2}B & \cdots & a_{nn}B \end{bmatrix}$$

- The vec$(\cdot)$ operation, which concatenates the columns of an $n \times m$ matrix $A = \{a_{ij}\}$ into an $nm \times 1$ vector can be defined as follows:

$$\text{vec}(A) = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{1n} & \cdots & \cdots & a_{1m} & \cdots & a_{nm} \end{bmatrix}^\top$$

Recall also the following results:

- For any matrix M, the matrices $M^\top M$ and $MM^\top$ are positive semi-definite.
- For a positive semi-definite matrix M, there exists a real matrix W such that $M = W^\top W$ and rank(W) = rank(M).
- All of the eigenvalues of a symmetric matrix M are real.
- All of the eigenvalues of a positive semi-definite matrix M are nonnegative.
- For a symmetric, nonnegative matrix M, there exists a nonnegative basis V for the invariant subspace associated with its Perron root $\rho(M)$.
- An orthogonal projector Π has the property that $\Pi^2 = \Pi$.
- For matrices A and B, $(A \otimes B)^\top = A^\top \otimes B^\top$.
- For matrices A, B, C, and D of compatible dimensions, $(A \otimes B)(C \otimes D) = AC \otimes BD$.
- If AXB is the product of three matrices A, X, and B, then vec$(AXB) = (B^\top \otimes A)vec(X)$.

A helpful reference for many general linear algebra results is Horn and Johnson [HJ85]. See Steeb's short text [S91] for a compilation of Kronecker product facts. A thorough source on graphs and hypergraphs is Berge's text [B73].

A final note: there exists a very rich and interesting subfield of linear algebra, called *spectral graph theory*, which is devoted to uncovering relationships between the spectral properties of matrix representations of graphs and their physical properties (e.g., connectivity, reducibility, etc.). Spectral methods are becoming increasingly attractive because they allow one to trade in combinatorial methods for faster and more efficient matrix methods. Often, algorithms that deal explicitly with all of the node and edge details of a graph are exponential in time and memory requirements, so the use of matrix approximation techniques is extremely helpful. An excellent introduction to spectral graph theory is given in Mohar in [M97], and a standard text is Chung [C97].

Chapter 3

Notions of similarity

AT THE END of Chapter 1, we briefly discussed some possible meanings of the term 'similarity' when applied to graphs. In this chapter, we will develop several of these notions more extensively. These approaches will provide a point of comparison for the similarity notion that is our primary focus, namely, the idea of *iterative similarity measures* presented in Chapter 4.

3.1 Isomorphism

If an isomorphism exists between two graphs, then they are structurally indistinguishable. More formally, two graphs are *isomorphic* if there exists a bijective (one-to-one and onto) function between the sets of nodes such that two nodes are connected in one graph if and only if their images under the bijection are connected. If graphs G_A and G_B are isomorphic, then there exists a permutation matrix, P, that will transform the adjacency matrix $\mathcal{A}$ into the adjacency matrix $\mathcal{B}$, i.e. $\mathcal{B} = P\mathcal{A}P^{\mathsf{T}}$. Additionally, if G_A and G_B are attributed graphs, then an isomorphism between them must also preserve the attributes on each node and edge. An example of two isomorphic graphs is given in Figures 3.1 and 3.2.

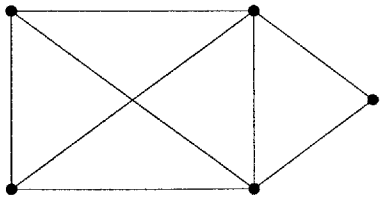


Figure 3.1: *An undirected graph.*

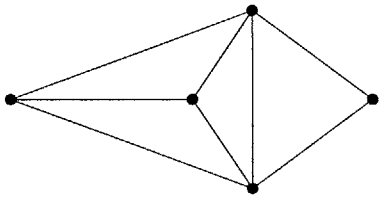


Figure 3.2: *A graph that is isomorphic to Figure 3.1.*

The complexity of determining whether two graphs are isomorphic is still undecided. Polynomial time algorithms have been developed for several special classes of graphs, including trees and planar graphs [T04],

but many existing algorithms, which often seek to find the correct matrix permutation, are exhaustive in the worst case: see, for example, [U76]. It is currently postulated that graph isomorphism may lie strictly between the P and NP-complete complexity classes [P99].

Observe that, because their adjacency matrices $\mathcal{A}$ and $\mathcal{B}$ can be made identical via a permutation, $\mathcal{A}$ and $\mathcal{B}$ must have the same set of eigenvalues; unfortunately, this is only a necessary condition. However, algorithms exist that use the spectra of two weighted graphs (of equal size) to find an approximation to a bijective function between each set of nodes that minimizes a certain error criterion; an isomorphism has been found when this function has zero error [U88].

A generalization of the graph isomorphism problem is that of *subgraph isomorphism*, in which one looks for isomorphic copies of a graph G_A *within* another graph G_B.

3.2 Error correcting graph matching

Often in pattern recognition problems, images will have missing or corrupted information that affects their representations as graphs. In this case, whole graph isomorphism will fail to recognize the underlying similarity of the images. One way to deal with corrupted information is to use an error correcting procedure. In its simplest form, the *error correcting graph matching* (or *edit distance*) between two graphs is the minimum number of *edit operations* (node and edge additions, substitutions and deletions) required to transform one graph into the other. Observe that this problem is a generalization of isomorphism; two graphs are isomorphic if their edit distance is zero.

Typically, different edit operations have different associated *costs* which are inversely related to how likely the edit is to occur. Thus, finding the edit distance between two graphs becomes an optimization problem: determine a minimum cost set of edit operations to transform one graph into another. Additionally, if the nodes and edges of the graph have their own attributes, then there may exist different costs for different pairs of substitutions. The optimal error correcting graph matching is highly dependent on the cost function; in [B99], Bunke presents results which describe whole classes of cost functions that will yield the same optimal matching.

3.3 Maximum common subgraph and minimum common supergraph

Two additional generalizations of subgraph isomorphism are the twin notions of *maximum common subgraph* and *minimum common supergraph*. These measures of graph similarity compare two subject graphs by generating a third. The maximum common subgraph of two graphs is the largest graph (by some measure involving the numbers of nodes and edges) contained in both subject graphs. Finding the maximum common subgraph of two graphs is another common way of dealing with loss or corruption of data in pattern recognition problems.

Similarly, the minimum common supergraph is the smallest graph that contains both subject graphs. In [BJK00], Bunke et al. observe that the problem of finding the minimum common supergraph is equivalent to identifying the maximum common subgraph, in the sense that algorithms for one can be trivially modified to find the other. Additionally, the authors find that, under a set of restrictions on the relationships between the costs of particular edit operations, there exists a lowest cost error-correcting graph matching that preserves the maximum common subgraph, G_S, and that the cost of this matching is an affine function of the size of G_S.

It is often useful to have a scalar measurement of the *distance* between two graphs, but graphs do not exist in an obvious metric space. However, the size of the maximum (minimum) common subgraph (supergraph) of two graphs can be used to define a metric distance between them: one such metric is presented by Bunke et al. in [BJK00] and another by Fernandez and Valiente in [FV01].

The maximum common subgraph problem is known to be NP-complete [GJ79], so most algorithms on general graphs are exhaustive in the worst case: see [L71] for an early but still influential example. In fact, finding the maximum common subgraph is equivalent to another graph theoretic problem. First, we'll define the *product graph*, G_{AB} between two graphs G_A and G_B. The node set of G_{AB} is $V_A \times V_B$ with an edge $((a,b),(c,d))$ existing if and only if (a,c) is an edge of G_A and (b,d) is an edge of G_B. Also, define a *clique* of G_{AB} as a subset of its vertices such that every vertex in the clique is connected to every other in the clique. Then finding the maximum common subgraph between G_A and G_B is equivalent to finding a maximum clique in the product graph G_{AB}.

In response to the complexity of this problem, many researchers have developed methods to approximate

the solution. These approximations typically involve embedding this discrete problem into a continuous space, then applying continuous optimization methods that are well-understood. In [P02], Pelillo develops an approximation algorithm for finding the maximum common subgraph of free trees by solving a continuous optimization problem with a quadratic cost function, using tools of evolutionary game theory.

3.4 Topological measures

For very large graphs, trying to compute the edit distance or the maximum common subgraph is a daunting task. One way of reducing large graphs to their essential characteristics is to represent them using aggregate topological measurements; for example, in- and out-degree distributions, clustering coefficients, betweenness measures and diameter, as discussed in Chapter 2.

Not only do these topological measures simplify the comparison of large graphs, but they often suggest a particular kind of growth process underlying the formation of the graph. Frequently, these growth processes are the primary interest of researchers; much of today's work in large graphs focuses on formulating a probabilistic model of graph evolution, then examining its statistics to see if they match what has been observed experimentally. In [JK01], Jain and Krishna present a weighted graph model representing the promotive or inhibitory effects of different chemical or biological species, then apply some simple rules of 'evolution' to the model and observe the spontaneous appearance of cooperativity and interdependence among the species.

One particular class of graphs that have received a lot of attention in the past few years are those that have *scale-free* degree distributions, i.e. the probability of any node having k neighbors is given by $k^{-\gamma}$ for some constant γ. The World Wide Web, Internet and many biological systems are all scale-free, albeit with different characteristic ranges for γ. As discussed in Section 1.1.2, the authors of [CLDG03] have developed a graph model for genetic evolution that matches the observed γ from experimental studies. An excellent overview of examples and approaches in complex networks is given by Albert and Barabási in [AB02].

Chapter 4

Iterative methods

THERE EXISTS a large class of similarity algorithms that take a very local perspective on similarity; namely, two nodes of two different graphs are considered 'similar' if their neighboring nodes are 'similar'. This is a cyclic definition, and very naturally leads to iterative updates, in which similarity scores between elements propagate along to neighboring elements at each time step. This chapter will explore several recent iterative approaches to computing node similarity scores between graphs, including several developed for specific applications.

4.1 Hub and authority scoring

An influential iterative approach to web searching was put forth by Kleinberg in [K99]. The author was motivated by the observation that traditional text-based search engines, which often naively rank search results via a function of the number of times a given query term appears in a page, routinely fail to find the most relevant pages. For example, the URL `http://www.mit.edu` might be the best response to a query on the term 'MIT', but the text 'MIT' may only appear once or not at all in the page.

A key insight is that the information content of the web is not simply the sum of the information within its individual webpages, but also includes the *relationships between pages*, i.e. the Web's link structure. Specifically, a link from page A to page B indicates that page A *endorses* page B in some way, or, as Kleinberg phrases it, A confers *authority* on B.

Kleinberg observes two different categories of results that might be relevant to a given query. The first category are pages he calls 'hubs', which point to many good sources of information on the query. A second category of relevent pages are the 'authorities', which contain detailed information regarding the query. Depending upon the objective of the user, one or both of these categories of results might be of interest.

Kleinberg then observes a cyclic relationship between these two categories of pages; a good hub is one that points to many good authorities, and a good authority is one that is pointed to by many good hubs.

This idea forms the core of Kleinberg's *Hypertext Induced Topic Selection* (HITS) search engine. First, a given query σ is fed into a text-based search engine to return a set of webpages V_{R_σ}. The link structure between elements of V_{R_σ} is mapped, yielding a graph $G_{R_\sigma} = G(V_{R_\sigma}, E_{R_\sigma})$. Next, G_{R_σ} is expanded to a larger graph $G_{S_\sigma} = G(V_{S_\sigma}, E_{S_\sigma})$ by adding a fixed number of in- and out-neighbors of each node in V_{R_σ}. G_{S_σ} is now the candidate graph whose pages will be scored and returned as the results of the search. These graphs are depicted in Figure 4.1. Denote the node-node adjacency matrix of G_{S_σ} by $\mathcal{A}_{S_\sigma}$.

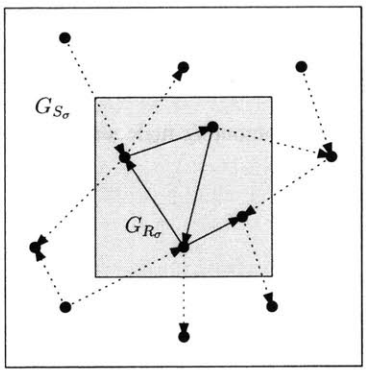


Figure 4.1: *The search result graph G_{R_σ} and the candidate graph G_{S_σ}.*

Each page n in V_{S_σ} is scored with a pair of numbers (h_n, a_n), where h_n is the *hub score* of page n and a_n is its *authority score*. As just discussed, these scores can be computed in a mutually reinforcing way. Kleinberg defines the HITS update procedure as follows: at stage k of the iteration,[1]

$$a_n(k) \leftarrow \sum_{i:(i,n)\in E_{S_\sigma}} h_i(k-1)$$

$$h_n(k) \leftarrow \sum_{i:(n,i)\in E_{S_\sigma}} a_i(k-1)$$

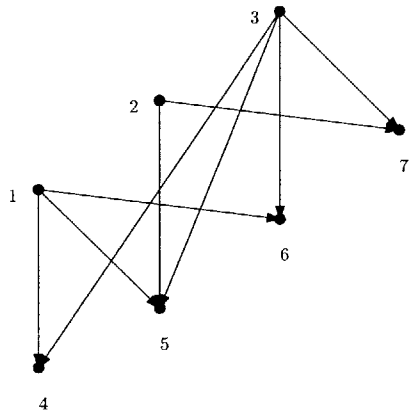
That is, the authority score of a node n at iteration k is the sum of the hub scores at iteration $k-1$ of the nodes that 'feed into' node n. Similarly, the hub score of node n at iteration k is the sum of the authority

[1] This is not the precise update formulated in [K99]. There, the author first computes $a_n(k)$ as defined here, then computes $h_n(k)$ using the most recent value $a_n(k)$ instead of $a_n(k-1)$ (i.e., a sequential rather than a simultaneous update). The results returned by these two approaches are identical - in Section 5.2.3, we will prove this statement in another context.

scores at iteration $k-1$ of the nodes that n 'feeds into'. Let us represent the set of hub scores at iteration k as a vector h_k and the authority scores as a vector a_k, and normalize h_k and a_k by some convenient vector norm at each stage (e.g., by the 2-norm of the vector, so that the sum of the squares of the entries of each vector equals 1). We can rewrite this update in matrix notation as follows:

$$\begin{bmatrix} h \\ a \end{bmatrix}_k \leftarrow \begin{bmatrix} 0 & \mathcal{A}_{S_\sigma} \\ \mathcal{A}_{S_\sigma}^T & 0 \end{bmatrix} \begin{bmatrix} h \\ a \end{bmatrix}_{k-1} \equiv M \begin{bmatrix} h \\ a \end{bmatrix}_{k-1}$$

where the normalization at each stage is necessary, but not explicitly indicated. Will this procedure converge? Observe that, since this matrix M is nonnegative and symmetric, all of its eigenvalues are real, and in particular $\rho(M)$ is one of the eigenvalues. This iteration is guaranteed to converge to a single limit as long as $-\rho(M)$ is not also an eigenvalue, an assumption that Kleinberg makes but which is not always valid in practice (see Section 4.2 for a discussion of this issue). Kleinberg also assumes that $\rho(M)$ has multiplicity 1, another assumption which is not always valid, and which has consequences for the magnitudes of the final scores. We will have much more to say on convergence issues in the following sections. An example of applying this iterative approach to a graph that can be clearly partitioned into hubs and authorities is given below in Figure 4.2.



nodes	hub score	authority score
1	0.374	0
2	0.242	0
3	0.467	0
4	0	0.365
5	0	0.467
6	0	0.365
7	0	0.308

Figure 4.2: *A graph and its hub and authority scores.*

Certainly, the hub and authority score results for the graph in Figure 4.2 finds the correct partitioning into 'hub-like' and 'authority-like' nodes. We also see that the node with the highest hub score points to the most nodes, and the node that is pointed to most has the highest authority score.

Given sets of hub and authority scores for all of the pages in V_{S_σ}, the HITS algorithm then returns the top scorers in each of the hub and authority categories. This method performs quite well, finding good pages that text-based search engines often miss, and allowing a user to choose which type of page is most relevant to him or her. For more details on implementation and performance results, refer to [K99]. For an interesting overview of the HITS approach and how it compares with other popular search engines, see [R04].

4.2 Blondel, Van Dooren, et.al, 2004

The work of Blondel, van Dooren et al. in [BVD04] is an interesting generalization of Kleinberg's approach. The authors begin with the recognition that, in essence, Kleinberg's algorithm compares each node in the candidate graph G_{S_σ} to the two nodes of a smaller *hub-authority graph* G_{HA}, depicted below in Figure 4.3. The hub and authority scores of each node in G_{S_σ} are similarity measures between each element in V_{S_σ} and the hub and authority elements in V_{HA}, respectively.

Figure 4.3: *The hub-authority graph HA.*

For a node i in G_{S_σ} and a node j in the hub-authority graph G_{HA}, let us denote their similarity by x_{ij} and define it as follows (again, with an implicit normalization at each iteration):

$$x_{ij}(k) \leftarrow \sum_{r:(r,i)\in E_{S_\sigma}, s:(s,j)\in E_{HA}} x_{rs}(k-1) + \sum_{r:(i,r)\in E_{S_\sigma}, s:(j,s)\in E_{HA}} x_{rs}(k-1) \tag{4.1}$$

Observe that, since there are no incoming edges to node h in G_{HA}, and no outgoing edges from node a, then for $j \in \{h, a\}$, we can simplify this equation for each of the two values of j:

$$x_{ih}(k) \leftarrow \sum_{r:(i,r)\in E_{S_\sigma}} x_{ra}(k-1)$$

$$x_{ia}(k) \leftarrow \sum_{r:(r,i)\in E_{S_\sigma}} x_{rh}(k-1)$$

which is precisely Kleinberg's update.

However, Eq. 4.1 is interesting on its own, for it gives us an update equation for a similarity measure between any two nodes in any two graphs, in which a node's score is updated in repsonse to the scores of its neighbors. In [BVD04], the authors present Eq. 4.1 as the update for two arbitrary graphs, G_A and G_B:

If we combine the scores x_{ij} into a $|V_A| \times |V_B|$ score matrix X, and if we use $\mathcal{A}$ and $\mathcal{B}$ to denote the node-node adjacency matrices of G_A and G_B, respectively, then this update at iteration k can be rewritten in matrix notation as follows:

$$X_k \leftarrow \mathcal{B}X_{k-1}\mathcal{A}^\top + \mathcal{B}^\top X_{k-1}\mathcal{A} \tag{4.2}$$

A third representation, which will be most useful, is obtained by applying the $vec(\cdot)$ operation to Eq. 4.2, which concatenates the columns of a matrix into one column vector. If we denote $vec(X_k) = x_k$, and recall the properties of this operation from Chapter 2, we obtain:

$$x_{k+1} \leftarrow \left(\mathcal{A} \otimes \mathcal{B} + \mathcal{A}^\top \otimes \mathcal{B}^\top\right) x_k \equiv M x_k$$

With this representation, determining the score vector x to which this procedure converges is a matrix iteration problem for which many tools exist. We will now state an important general result, the proof of which is based on the proof of Theorem 2 in [BVD04]:

Theorem 4.2.1. *Let M be a symmetric, non-negative matrix with Perron root ρ. Let $x_0 > 0$ and define:*

$$x_{k+1} = \frac{Mx_k}{\| Mx_k \|_2}$$

When $-\rho$ is not an eigenvalue of M, then the sequence x_k converges to $\Pi x_0 / \| \Pi x_0 \|_2$, where Π is the orthogonal projector onto the invariant subspace associated with ρ. If $-\rho$ is an eigenvalue of M, then the odd and even subsequences x_{2k} and x_{2k+1} converge to the following limits:

$$x_{even} = \frac{\Pi x_0}{\| \Pi x_0 \|_2}$$

$$x_{odd} = \frac{\Pi M x_0}{\| \Pi M x_0 \|_2}$$

where Π *is the orthogonal projector onto the sum of the invariant subspaces associated with* ρ *and* $-\rho$.

Proof. Let V_ρ be a matrix whose columns are orthonormal vectors that form a basis for the invariant subspace of M associated with eigenvalue ρ, and let $V_{-\rho}$ be similarly defined for the invariant subspace associated with the eigenvalue $-\rho$. (If $-\rho$ is not an eigenvalue of M, then $V_{-\rho}$ is zero). Let V_μ be an orthonormal matrix whose columns form a basis for the invariant subspace associated with the rest of the spectrum of M. Then $MV_\rho = \rho V_\rho$, $MV_{-\rho} = -\rho V_{-\rho}$ and $MV_\mu = V_\mu M_\mu$ for some square matrix M_μ whose largest eigenvalue has magnitude $\mu < \rho$. Then we can decompose M using its eigenvalues and eigenvectors as:

$$M = \rho V_\rho V_\rho^\top - \rho V_{-\rho} V_{-\rho}^\top + V_\mu M_\mu V_\mu^\top$$

Since V_ρ, $V_{-\rho}$ and V_μ have mutually orthogonal columns,

$$M^2 = \rho^2 (V_\rho V_\rho^\top + V_{-\rho} V_{-\rho}^\top) + V_\mu M_\mu^2 V_\mu^\top \equiv \rho^2 \Pi + V_\mu M_\mu^2 V_\mu^\top$$

Clearly then, for any even power of M,

$$M^{2k} = \rho^{2k} \Pi + V_\mu M_\mu^{2k} V_\mu^\top$$

Now, we will show that $\Pi x_0 > 0$. First, observe that Π is the projector onto the invariant subspace associated with the eigenvalue ρ^2 of the matrix M^2. Since M is symmetric and nonnegative, so is M^2. Thus, there exists a nonnegative basis for the invariant subspace associated with the eigenvalue ρ^2 of M^2, and therefore a nonnegative projector $\tilde{\Pi}$. Thus, for a nonnegative $\tilde{\Pi}$ and a positive x_0, $\tilde{\Pi} x_0 > 0$. Since Π and $\tilde{\Pi}$ project onto the same subspace in $\Re^n$, $\Pi x_0 = \tilde{\Pi} x_0$, which means that $\Pi x_0 > 0$. Similarly, $\Pi M x_0 > 0$; since M is nonnegative, $M x_0$ must be strictly greater than zero and the same argument applies.

Now, observe that the maximum amplification that the matrix $V_\mu M_\mu V_\mu^\top$ can exert on a vector is $\mu < \rho$.

Thus, if we multiply M^{2k} by x_0 and normalize, the result is:

$$\begin{aligned} x_{2k} &= \frac{\left(\rho^{2k}\Pi + V_\mu M_\mu^{2k} V_\mu^\top\right) x_0}{\| \left(\rho^{2k}\Pi + V_\mu M_\mu^{2k} V_\mu^\top\right) x_0 \|_2} = \frac{\left(\Pi + \frac{1}{\rho^{2k}} V_\mu M_\mu^{2k} V_\mu^\top\right) x_0}{\| \left(\Pi + \frac{1}{\rho^{2k}} V_\mu M_\mu^{2k} V_\mu^\top\right) x_0 \|_2} \\ &= \frac{\Pi x_0}{\| \Pi x_0 \|_2} + O(\mu/\rho)^{2k} \end{aligned}$$

Taking the limit as $k \to \infty$:

$$x_{even} = \lim_{k\to\infty} \frac{\Pi x_0}{\| \Pi x_0 \|_2} + O(\mu/\rho)^{2k} = \frac{\Pi x_0}{\| \Pi x_0 \|_2}$$

We obtain a similar result for the odd limit:

$$x_{2k+1} = \frac{\Pi M x_0}{\| \Pi M x_0 \|_2} + O(\mu/\rho)^{2k+1}$$

Therefore,

$$x_{odd} = \lim_{k\to\infty} \frac{\Pi M x_0}{\| \Pi M x_0 \|_2} + O(\mu/\rho)^{2k+1} = \frac{\Pi M x_0}{\| \Pi M x_0 \|_2}$$

□

Note that if ρ has multiplicity 1, and if $-\rho$ is not an eigenvalue, then the choice of initialization of x_0 is irrelevent; the process will converge to the unique dominant eigenvector of M. If, however, ρ is associated with more than one eigenvector or $-\rho$ is an eigenvalue, the choice of initial condition is critical to the final score. Bearing these observations in mind, the authors of [BVD04] choose to initialize x_0 to the all-ones vector, and choose their final score to be the limit of the *even* iterations. An example of the application of this similarity scoring method is given in Figure 4.4 and Table 4.1

4.3 Application-based approaches

Several other application-oriented algorithms emerged at roughly the same time as [BVD04] that utilize the idea of recursively computing node similarity scores based on the scores of neighboring nodes.

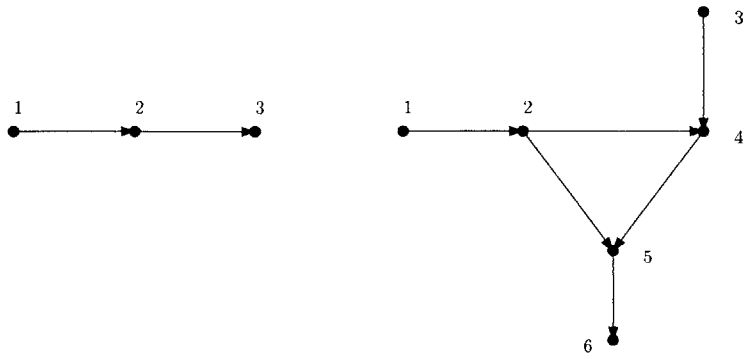


Figure 4.4: *Two directed graphs.*

nodes	1	2	3
1	0.211	0	0
2	0.426	0.336	0
3	0.255	0.096	0
4	0.171	0.406	0.271
5	0	0.271	0.466
6	0	0	0.171

Table 4.1: *[BVD04] node similarity scores for the graphs in Figure 4.4.*

4.3.1 Similarity flooding

The goal of the similarity flooding algorithm presented by Melnik, Garcia-Molina and Rahm in [MGMR02] is the matching of two different database structures, D_A and D_B. For example, if one were combining two personnel databases from two merging companies, one might want to be able to match the field 'Emp. Name' in one database with the field 'Name' in the second. To perform this matching, the authors take an approach that is similar to Kleinberg's hub-authority method. First, a preliminary similarity score between the *text labels* associated with each element in the database is computed according to some predefined function, without considering the database structure. Elements in a database might be any kind of data structure, including tables or individual fields. Next, each database is transformed into a representative graph (G_A and G_B, respectively) with labels on the nodes (database elements) and edges (relationships between elements). A particular method of performing this transformation is not defined, but examples are given in which node P points to node Q with edge label 'column' if Q is a daughter column in table P.

Next, some heuristically-derived weight function is defined: $w(a_1, a_2), (b_1, b_2))$ for $(a_1, a_2) \in E_A$ and $(b_1, b_2) \in E_B$. This function compares edges drawn from G_A and G_B, respectively, and assigns a 'matching'

value. The update procedure of [MGMR02] is then:

$$x_{ij} \leftarrow x_{ij} + \sum_{r:(r,i)\in E_B, s:(s,j)\in E_A} w((r,i),(s,j))x_{rs} + \sum_{r:(i,r)\in E_B, s:(j,s)\in E_A} w((r,i),(s,j))x_{rs}$$

with normalization at each stage by the maximum x_{ij}. Once a matrix of similarity scores is obtained, the authors describe several filters that can be applied to this data to obtain a one-to-one mapping between elements in each database, including solving a stable marriage problem or simply maximizing the sum of matched scores. We will discuss matching issues in more detail in Chapter 6.

Because this approach has several arbitrary functions embedded inside it, its properties are not particularly easy to analyze, but the authors report positive results in matching performance as compared to a benchmark of human matchers.

4.3.2 SimRank

The SimRank algorithm, described by Jeh and Widom in [JW02], computes the similarities between pairs of nodes within a single graph; that is, the algorithm computes a graph's *self-similarity*. This kind of information has applications in recommender systems, where prior similar purchases on the parts of two consumers leads one to infer the consumers' similarity. In light of this motivation, [JW02] focuses on the influence of *incoming edges* to each node in its similarity score. This update procedure is:

$$x_{ij} \leftarrow \sum_{r:(r,i)\in E_B, s:(s,j)\in E_A} x_{rs}$$

The authors also describe variations on their approach, including using non-linear functions within the update and, citing [K99], defining a separate score that counts the influence of *outgoing edges* as well.

4.3.3 Heymans and Singh, 2003

As evidence of the broad applicability of iterative methods, in [HS03], Heymans and Singh present an interesting challenge: comparing a set of graphs, each of which represent enzyme interactions within a particular species, to determine the evolutionary relationship between the species. The evolutionary relationship they

seek to construct is a *phylogenetic tree*, analogous to a family tree in the sense that every branch of the tree represents descendents from a common ancester. To do this, the *enzyme graphs* of many species are compared pairwise, each yielding a node similarity matrix. As described in Chapter 1, enzyme graphs are a particular mapping of metabolic networks; enzymes serve as nodes in the graph, with an edge existing from enzyme e_1 to enzyme e_2 if e_1 catalyzes a reaction whose product is the substrate for e_2. The pairwise similarity matrices are computed, then used to compute a *distance* between the two species. If N species are compared this way, the result is an $N \times N$ matrix of distances which can then be fed into a software program for computing phylogenetic trees that maximizes certain performance objectives.

To initialize their method, the authors compare the enzymes represented by each node using their four-digit Enzyme Commission numbers, which categorize the type of reaction that the enzyme catalyzes, yielding a number between 0 and 1. What is perhaps most interesting about the method of [HS03] are the terms used in the update. Not only does their update include the positive flow of similarity from neighboring nodes, but also includes positive flows from nodes that are *not connected*, as well as penalizing flows from neighbors that are mismatched. The simplest way to see this is to examine their update equation.

Define I_B to be the matrix of all ones with the same dimension as $\mathcal{B}$. Removing many constants, extraneous details and scaling terms, a stripped-down version of their update is as follows, accompanied by text describing the contributions to a given similarity score x_{ij}:

X	$\leftarrow \mathcal{B}X\mathcal{A}^\top + \mathcal{B}^\top X\mathcal{A}$	[BVD04] terms, summing the scores of pairs (p,q) in which $(p,j) \in E_B, (q,i) \in E_A$ or $(j,p) \in E_B, (i,q) \in E_A$.
	$+(I_B - \mathcal{B})X(I_A - \mathcal{A})^\top + (I_B - \mathcal{B})^\top X(I_A - \mathcal{A})$	summing the scores of pairs (p,q) in which $(p,j) \notin E_B, (q,i) \notin E_A$ or $(j,p) \notin E_B, (i,q) \notin E_A$.
	$-\mathcal{B}X(I_A - \mathcal{A})^\top - \mathcal{B}^\top X(I_A - \mathcal{A})$	subtracting the scores of pairs (p,q) in which $(p,j) \in E_B, (q,i) \notin E_A$ or $(j,p) \in E_B, (i,q) \notin E_A$.
	$-(I_B - \mathcal{B})X\mathcal{A}^\top - (I_B - \mathcal{B})^\top X\mathcal{A}$	subtracting the scores of pairs (p,q) in which $(p,j) \notin E_B, (q,i) \in E_A$ or $(j,p) \notin E_B, (i,q) \in E_A$.

The justification for the use of the extra terms in this update is that, in an evolutionary setting, the absence of connections between nodes can be as meaningful as the presence of such connections. Additionally, the more connections that one enzyme graph has that the other doesn't indicates that the two have probably not

descended from the same species recently, and thus should reduce their similarity.

The inclusion of the subtracted terms means that we can end up with a negative similarity score between two nodes. When a vectorization is applied to the matrix X, transforming it to the column vector x, the resulting expression is:

$$\begin{aligned} x \leftarrow\ & [\mathcal{A} \otimes \mathcal{B} + \mathcal{A}^\top \otimes \mathcal{B}^\top + (I_A - \mathcal{A}) \otimes (I_B - \mathcal{B}) + (I_A - \mathcal{A})^\top \otimes (I_B - \mathcal{B})^\top \\ & -(I_A - \mathcal{A}) \otimes \mathcal{B} - (I_A - \mathcal{A})^\top \otimes \mathcal{B}^\top - \mathcal{A} \otimes (I_B - \mathcal{B}) - \mathcal{A}^\top \otimes (I_B - \mathcal{B})^\top]x \end{aligned}$$

This update matrix is symmetric, so its eigenvalues are purely real. Thus, the normalized even and odd iterations will each converge to a real vector of scores.

The authors apply this approach to several sets of data from varying numbers of species, and report positive results in comparing their phylogenetic trees with those that are already well understood. Some numerical examples of this similarity measure are given in Section 5.3.

Chapter 5

Coupled node-edge scoring

ALL OF the iterative similarity methods in the previous chapter return a set of *node similarity scores*, without considering the possibility of also scoring the similarity of *edges*. This might be a useful notion in graphs where the relationships represented by the edges are also important. Using the iterative method framework, in which two graph elements are similar if their neighborhoods are similar, we can immediate suggest one possible way to constructing an edge score: *an edge in G_B is like an edge in G_A if their source and terminal nodes are similar, respectively.*

This definition of edge similarity introduces a coupling between edge scores and node scores, which could be interesting. Thus, our approach will be to update the edge scores via the node scores and vice versa.

5.1 Constructing a coupled node-edge score update

First, arbitrarily number the edges and nodes of two graphs, G_A and G_B. Denote the number of nodes and edges in G_A by n and m, respectively, and the number of nodes and edges in G_B by q and p, respectively. 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 in graph G_A (similarly for G_B). Also, let x_{ij} denote the node score between node i in G_B and node j in G_A, and y_{ij} the edge score between edge i in G_B and edge j in G_A. An update equation for node and edge scores takes the following form:

$$y_{ij}(k) \leftarrow x_{s(i)s(j)}(k-1) + x_{t(i)t(j)}(k-1) \tag{5.1}$$

$$x_{ij}(k) \leftarrow \sum_{t(k)=i,t(l)=j} y_{kl}(k-1) + \sum_{s(k)=i,s(l)=j} y_{kl}(k-1) \tag{5.2}$$

These scores can be assembled into matrices: $Y = \{y_{ij}\}$ and $X = \{x_{ij}\}$. A normalization at each stage by

the Frobenius norm (or any appropriate matrix norm) is required, but will not be explicitly represented when discussing the update unless it is relevant to the work at hand.

To begin to work with such a definition, both the nodes and edges of each graph must be identified and enumerated. Using the construction of edge-source matrices A_S and edge-terminus matrices A_T of a graph G_A as defined in Section 2.3, the update process represented in Eqs. 5.1 and 5.2 can be written concisely in the following matrix form:

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

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

Following the development in [BVD04] that was discussed in Section 4.2, we can vectorize these equations. If we denote $y = \mathrm{vec}(Y)$ and $x = \mathrm{vec}(X)$, the update becomes:

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

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

Concatenating these vectors yields a single matrix equation:

$$\begin{bmatrix} x \\ y \end{bmatrix}_k \leftarrow \begin{bmatrix} 0 & G^\top \\ G & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}_{k-1} \equiv M \begin{bmatrix} x \\ y \end{bmatrix}_{k-1} \tag{5.3}$$

5.2 Properties of the coupled model

In this section, we will explore some of the interesting features of this coupled model.

5.2.1 Invariance to even/odd iteration

Observe that if $\begin{bmatrix} a^\top & b^\top \end{bmatrix}^\top$ is a right eigenvector of M as defined in Eq. 5.3 associated with the eigenvalue λ, then $\begin{bmatrix} -a^\top & b^\top \end{bmatrix}^\top$ is an eigenvector of M associated with the eigenvalue $-\lambda$. Thus, eigenvalues of M always

have their complement as an eigenvalue.

As was stated in Theorem 4.2, this implies that there will exist different limits for the odd and even subsequences of s_k. To see what might happen, observe that if $\begin{bmatrix} x^\top & y^\top \end{bmatrix}^\top$ is an eigenvector of M with eigenvalue 1, then the following equation holds:

$$\begin{bmatrix} 0 & G^\top \\ G & 0 \end{bmatrix} \begin{bmatrix} ax \\ by \end{bmatrix} = \begin{bmatrix} bx \\ ay \end{bmatrix}$$

Clearly, iterating this procedure will cause the score vector s to jump back and forth between two different values. However, recall that x and y are two separate sets of scores that we concatenated into a single vector for convenience. Thus, it is irrelevent to us if the constants that multiply each of these score sets changes at every iteration. Since we can normalize x and y however we'd like, the normalized even and odd limits of this process are essentially the same. In fact, if ρ denotes the Perron root of M, the relationship between the two is precisely:

$$s_{even} = \begin{bmatrix} x \\ y \end{bmatrix}, s_{odd} = \begin{bmatrix} \alpha\rho x \\ \frac{1}{\alpha\rho} y \end{bmatrix}.$$

Without loss of generality, assume that ρ is a unique eigenvalue (which implies that its complement is also). The consequence of this assumption is that the invariant subspace associated with ρ has dimension 1 (i.e., it is the span of the unique right eigenvector of M associated with eigenvalue ρ). If this assumption does not hold and the invariant subspace associated with ρ has higher dimension, it simply means carrying extra terms through the computation, but the result is identical. Denote the unit length right eigenvector of M associated with eigenvalue ρ by $\begin{bmatrix} a^\top & b^\top \end{bmatrix}$. Note that this implies that:

$$\begin{bmatrix} 0 & G^\top \\ G & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} G^\top b \\ Ga \end{bmatrix} = \begin{bmatrix} \rho a \\ \rho b \end{bmatrix}$$

As presented in Theorem 4.2, the limit s_{even} is

$$s_{even} = \frac{\Pi s(0)}{\| \Pi s(0) \|_2}$$

where Π is a projector onto the invariant subspace associated with the eigenvalues $\pm\rho$. Thus, Π can be written as:

$$\Pi = \begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} a^\top & b^\top \end{bmatrix} + \begin{bmatrix} -a \\ b \end{bmatrix} \begin{bmatrix} -a^\top & b^\top \end{bmatrix}$$

If $s_0 = \begin{bmatrix} x_0^\top & \alpha G x_0^\top \end{bmatrix}^\top$, then:

$$\begin{aligned} \Pi s_0 &= \left(\begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} a^\top & b^\top \end{bmatrix} + \begin{bmatrix} -a \\ b \end{bmatrix} \begin{bmatrix} -a^\top & b^\top \end{bmatrix} \right) \begin{bmatrix} x_0^\top & \alpha G x_0^\top \end{bmatrix}^\top \\ &= \begin{bmatrix} a(a^\top + \alpha b^\top G)x_0 - a(-a^\top + \alpha b^\top G)x_0 \\ b(a^\top + \alpha b^\top G)x_0 + b(-a^\top + \alpha b^\top G)x_0 \end{bmatrix} \\ &= \begin{bmatrix} 2(aa^\top)x_0 \\ 2\alpha(\alpha b b^\top G)x_0 \end{bmatrix} \\ &= \begin{bmatrix} 2(aa^\top)x_0 \\ 2\alpha\rho(ba^\top)x_0 \end{bmatrix} \equiv \begin{bmatrix} \gamma \\ \delta \end{bmatrix} \end{aligned}$$

In the odd limit,

$$s_{odd} = \frac{\Pi M s(0)}{\| \Pi M s(0) \|_2}$$

Following a similar set of calculations, we find that

$$\Pi M s_0 = \begin{bmatrix} 2\alpha\rho^2(aa^\top)x_0 \\ 2\rho(ba^\top)x_0 \end{bmatrix} = \begin{bmatrix} \alpha\rho^2\gamma \\ \frac{1}{\alpha}\delta \end{bmatrix}$$

Now, normalizing as prescribed:

$$s_{even} = \begin{bmatrix} \gamma \\ \delta \end{bmatrix} \frac{1}{\| \begin{bmatrix} \gamma^\top & \delta^\top \end{bmatrix}^\top \|_2}$$

$$s_{odd} = \begin{bmatrix} \alpha\rho^2\gamma \\ \frac{1}{\alpha}\delta \end{bmatrix} \frac{1}{\| \begin{bmatrix} \alpha\rho^2\gamma^\top & \frac{1}{\alpha}\delta^\top \end{bmatrix}^\top \|_2} = \begin{bmatrix} \alpha\rho\gamma \\ \frac{1}{\alpha\rho}\delta \end{bmatrix} \frac{1}{\| \begin{bmatrix} \alpha\rho\gamma^\top & \frac{1}{\alpha\rho}\delta^\top \end{bmatrix}^\top \|_2}$$

We can also observe this property by using the block off-diagonal structure to *decouple* the computations of x_k and y_k. Initializing x and y to x_0 and y_0, respectively, we can write explicit formulas for x_k and y_k:

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

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

Defining $y_0 = \alpha G x_0$, where $\alpha > 0$ is some constant:

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

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

It is important here to recall that each step in the iteration is normalized by some norm of the score vector. Thus, the multiplication by $\frac{1}{\alpha}$ in the even iterations of x and the odd iterations of y will be normalized away and have no bearing on the final score.

Now, recall Theorem 4.2, which describes the limiting behavior of this kind of matrix iteration. Since $G^\top G$ is positive semi-definite, its eigenvalues are all nonnegative. Thus, it is not possible for $-\rho(G^\top G)$ to be an eigenvalue of $G^\top G$ and so the even and odd iterations of the computation of x converge to the same

value, namely the orthogonal projection of $G^\top y_0$ onto the invariant subspace associated with the Perron root $\rho(G^\top G)$. An analogous result holds for the iterations of y, since $GG^\top$ is also positive semi-definite.

5.2.2 Choice of initial condition

Interestingly, if x_0 is chosen to be the all-ones vector, then $Gx_0 = 2y_0$, where y_0 is also an all-ones vector. To see this, set x_0 to be the $nq \times 1$ vector of ones, $\vec{1}$. The product Gx_0 is an $mp \times 1$ column vector containing the row sums of G. Since $(Gx_0)^\top = x_0^\top G^\top$, which is a vector containing the column sums of $G^\top$. Recall that $G^\top = A_S \otimes B_S + A_T \otimes B_T$.

Now we will argue that each column sum of $G^\top$ is equal to 2. To see this, examine the first column of $A_S \otimes B_S$. The first column of A_S contains exactly one 1, as does the first column of B_S - thus, there will be a single '1' in the first column of $A_S \otimes B_S$. Since this is true for any column of $A_S \otimes B_S$, we see that the column sums of this matrix are all 1. An identical argument applies to $A_T \otimes B_T$. Thus, each column sum in $G^\top$ is equal to 2, which means that every row sum in G is also equal to 2 and $G\vec{1} = 2\vec{1}$, where any occurrence of $\vec{1}$ has the appropriate dimensions.

Recall that the all-ones vector was the initial condition used in the iteration of [BVD04]; thus, we see that it is also a natural choice for this coupled method. It also has some intuitive merit, since there is no reason *a priori* to expect that any node or edge in one graph is more or less like any node or edge in another graph.

5.2.3 Sequential updating

The iteration that we've been working with has updated the node and edge scores *simultaneously*. Now, consider a sequential update that could be defined as:

$$y_k = Gx_{k-1} \tag{5.4}$$

$$x_k = G^\top y_k \tag{5.5}$$

or as:

$$x_k = G^\top y_{k-1}$$

$$y_k = G x_k$$

Will this procedure give us the same result as the simultaneous update? The answer is yes, and can be easily shown. Let us consider the first kind of sequential update represented in Eqs. 5.4 and 5.5. Then a matrix representation for this update is:

$$\begin{bmatrix} x \\ y \end{bmatrix}_k = \begin{bmatrix} 0 & G^\top \\ 0 & GG^\top \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}_{k-1} \equiv N \begin{bmatrix} x \\ y \end{bmatrix}_{k-1}$$

Observe that if $\begin{bmatrix} a^\top & b^\top \end{bmatrix}^\top$ is a right eigenvector of M (as defined in the simultaneous update) associated with eigenvalue λ, then $\begin{bmatrix} \frac{1}{\lambda}a^\top & b^\top \end{bmatrix}^\top$ is a right eigenvector of N associated with eigenvalue λ^2. To see this:

$$\begin{bmatrix} 0 & G^\top \\ 0 & GG^\top \end{bmatrix} \begin{bmatrix} a/\lambda \\ b \end{bmatrix} = \begin{bmatrix} G^\top b \\ GG^\top b \end{bmatrix} = \begin{bmatrix} \lambda a \\ \lambda^2 b \end{bmatrix} = \lambda^2 \begin{bmatrix} a/\lambda \\ b \end{bmatrix}$$

Thus, if the simultaneous update converges to scores x and y, the sequential update will converge to the same set of scores.

5.2.4 Expanding the decoupled updates

We can take the decoupling described in Section 5.2.1 one step further by applying the definition of G. First, we'll focus on the node similarity vector x.

$$\begin{aligned} x_k &\leftarrow (G^\top G) x_{k-1} \\ &= (A_S \otimes B_S + A_T \otimes B_T)\left(A_S^\top \otimes B_S^\top + A_T^\top \otimes B_T^\top\right) x_{k-1} \\ &= \left(A_S A_T^\top \otimes B_S B_T^\top + A_T A_S^\top \otimes B_T B_S^\top + A_S A_S^\top \otimes B_S B_S^\top + A_T A_T^\top \otimes B_T B_T^\top\right) x_{k-1} \\ &= \left(\mathcal{A} \otimes \mathcal{B} + \mathcal{A}^\top \otimes \mathcal{B}^\top + D_{A_S} \otimes D_{B_S} + D_{A_T} \otimes D_{B_T}\right) x_{k-1} \end{aligned}$$

where $\mathcal{A}$, D_{A_S} and D_{A_T} are defined in Section 2.3. Reshaping the vector x into its matrix form, X, we obtain:

$$X_k \leftarrow \mathcal{B}X_{k-1}\mathcal{A}^\top + \mathcal{B}^\top X_{k-1}\mathcal{A}^\top + D_{B_S}X_{k-1}D_{A_S} + D_{B_T}X_{k-1}D_{A_T}.$$

Explaining this update is not difficult. First, denote the *out-degree* of node i by $O(i)$ and the in-degree by $I(i)$. Then, to obtain a new x_{ij}, multiply the existing x_{ij} by the quantity $O(i)O(j) + I(i)I(j)$, then add the scores of the neighboring nodes. Interestingly, this is not too different from the approach taken by the similarity flooding algorithm put forth in [MGMR02]. Recall from Section 4.3.1 that the update in this paper is:

$$x_{ij} \leftarrow x_{ij} + \sum_{r:(r,i)\in E_B, s:(s,j)\in E_A} w((r,i),(s,j))x_{rs} + \sum_{r:(i,r)\in E_B, s:(j,s)\in E_A} w((r,i),(s,j))x_{rs}$$

In our approach, there is a multiplicative term before the x_{ij} and an identity weighting function.

Also, recall that the iteration in [BVD04] is:

$$X_k \leftarrow \mathcal{B}X_{k-1}\mathcal{A}^\top + \mathcal{B}^\top X_{k-1}\mathcal{A}^\top$$

Thus, our node calculation differs from that in [BVD04] by the inclusion of additional diagonal terms that serve to amplify the scores of nodes that are highly connected.

We can also expand the expression for the edge similarity vector y:

$$\begin{aligned} y_k &\leftarrow (GG^\top)y_{k-1} \\ &= \left(A_S^\top \otimes B_S^\top + A_T^\top \otimes B_T^\top\right)(A_S \otimes B_S + A_T \otimes B_T)\, y_{k-1} \\ &= \left(A_T^\top A_S \otimes B_T^\top B_S + A_S^\top A_T \otimes B_S^\top B_T + A_S^\top A_S \otimes B_S^\top B_S + A_T^\top A_T \otimes B_T^\top B_T\right) y_{k-1} \end{aligned}$$

Recall the properties of these matrices, given in Section 2.3. In matrix form:

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

5.2.5 Interpreting node similarity scores

Recall the definition of a *product graph* given in Section 3.3. The product graph G_{AB} between two graphs G_A and G_B has node set $V_A \times V_B$ with an edge $((a,b),(c,d))$ existing if and only if (a,c) is an edge of G_A and (b,d) is an edge of G_B. It is not difficult to see that the adjacency matrix of G_{AB} will be $\mathcal{A} \otimes \mathcal{B}$.

In the coupled score model, node similarity scores are updated as follows:

$$x_{k-1} \leftarrow \left(\mathcal{A} \otimes \mathcal{B} + \mathcal{A}^\top \otimes \mathcal{B}^\top + D_{A_S} \otimes D_{B_S} + D_{A_T} \otimes D_{B_T}\right) x_k \tag{5.6}$$

In light of this observation, we can interpret the update matrix in Eq. 5.6 as an *undirected* adjacency matrix of the product graph of G_A and G_B, with the diagonal terms contributing multiple self-loops at every non-singleton product node. Singleton nodes ij in this product graph will arise when node i in G_A has no outgoing edges and node j in G_B has no incoming edges, or vice versa. Thus, we can regard our node similarity score update procedure as the propagation of scores along the edges of this product graph at each time step, with each node summing the scores of its neighbors.

5.2.6 Self-similarity

Often, one is interested in the similarity scores between a given graph and *itself*. We'll refer to these scores as the *self-similarity* of a graph. Given positive semi-definite initial conditions X_0 and Y_0, the node and edge self-similarity matrices of a graph are also positive semi-definite. To see this, we can use the decoupled description of these updates given in the previous section. The node self-similarity update for a graph G_A is:

$$X_k \leftarrow \mathcal{A} X_{k-1} \mathcal{A}^\top + \mathcal{A}^\top X_{k-1} \mathcal{A} + D_{A_S} X_{k-1} D_{A_S} + D_{A_T} X_{k-1} D_{A_T}$$

.

If X_0 is positive semi-definite, then it can be decomposed into a product $X_0 = WW^\top$ where W is non-singular. Then for some vector v,

$$v^\top \mathcal{A} X_0 \mathcal{A}^\top v = v \mathcal{A} W W^\top \mathcal{A}^\top v^\top = < W^\top \mathcal{A}^\top v, W^\top \mathcal{A}^\top v > \geq 0$$

Similarly, $v^\top \mathcal{A}^\top X_0 \mathcal{A} v \geq 0$. Thus, the node score update is a scaled sum of two positive semi-definite matrices, which is also positive semi-definite. A parallel result holds for edge self-similarity scores. Observe that the all-ones intial conditions, $X_0 = \vec{1}\vec{1}^\top$ and $Y_0 = \vec{1}\vec{1}^\top$ are both positive semi-definite. We will use these results in Chapter 6 in our discussion of identifying isomorphic graphs.

5.2.7 Paths and cycles

We've mostly focused on adjacency matrices with entries that are '0' or '1'; either an edge exists between two nodes, or it does not. It's entirely possible to assign *weights* to edges by using values in the interval [0, 1]. As one example of this, consider the graph with a single weighted edge depicted in Figure 5.1; the nodes of this graph are labeled '1' through '3', and the edge weights are given as shown.

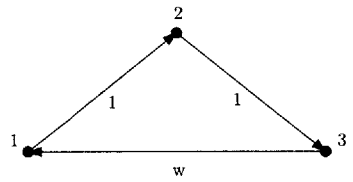


Figure 5.1: *A graph with weighted edges.*

When $w = 0$, this is simply a path graph on three nodes. When $w = 1$, this is a cycle graph. It is interesting to observe how the node self-similarity scores change as w is varied from 0 to 1 and beyond. This result is depicted in Figure 5.2, where each line depicts one of the entries in the self-similarity matrix.

The simple nature of this graph makes it easy to compute the trajectory of similarity scores as w is varied by computing the dominant eigenvector(s) of $G^\top G$. For $w = 1$, the dominant eigenvalue is associated with three eigenvectors, v_1, v_2 and v_3:

$$v_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$v_3 = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The similarity score is a result of the projection of the all-ones vector onto the subspace that they span,

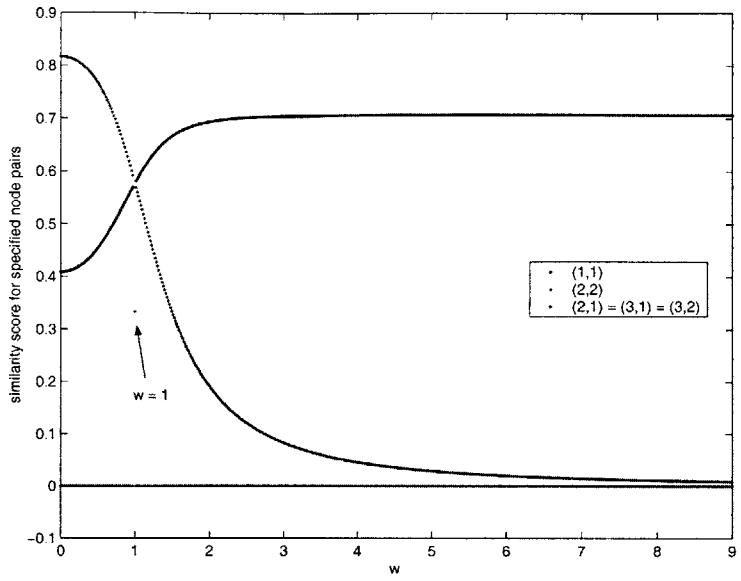


Figure 5.2: *Node self-similarity scores for the graph of Figure 5.1 as a function of the weight w.*

yielding the following:

$$X \propto \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

For $w \neq 1$, the matrix has a single dominant eigenvector, and thus the node similarity score matrix, X will be as follows:

$$X \propto \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1-2w^2+\sqrt{9-4w^2+4w^4}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For $w = 0$, the node scores are:

$$X \propto \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

confirming the result given in Figure 5.2. As $w \to \infty$, this matrix becomes:

$$X \propto \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Interestingly, this is the same self-similarity matrix obtained for the graph depicted in Figure 5.3, suggesting that the as $w \to \infty$, the other edges cease to be relevant.

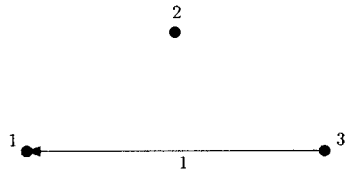


Figure 5.3: *The self similarity scores of this graph are the same as those for the graph in Figure 5.1 in the limit as* $w \to \infty$.

Note that the method used to compute the node similarity scores here is critical; for w close to 1, the second largest eigenvalue of the matrix $G^T G$ approaches the dominant eigenvalue, and thus the iteration will take infinitely long to converge.

5.2.8 Similarity scores of projective planes

One interesting class of graphs is the set that represent *finite projective planes*. A finite projective plane of order n is defined as a set of $n^2 + n + 1$ *points* and $n^2 + n + 1$ *lines* with the following incidence properties [L91]:

1. Every line contains $n + 1$ points.
2. Every point is on $n + 1$ lines.
3. Any two distinct lines intersect at exactly one point.
4. Any two distinct points lie on exactly one line.

There are two ways to represent a projective plane as a graph. First, we might use the *points* and *lines* of the projective plane to draw *vertices* and *edges*, respectively. An example of this construction is given below

in Figure 5.4 for the Fano plane, the unique projective plane of order 2..

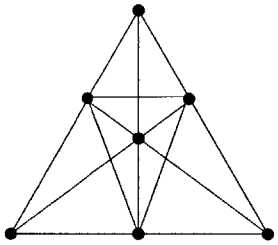


Figure 5.4: *One representation of the Fano plane, a projective plane of order 2.*

A second way of representing a projective plane of order n is as a graph in which one set of $n^2 + n + 1$ vertices represents *points* and a second equally-sized set represents *lines*. An edge then connects the vertices representing point i and point j if i is on line j. These graphs are referred to as the *point-line incidence graphs* of the projective plane. Observe that this graph is an example of a *bipartite graph*, in which the node set can be partitioned into two subsets A and B such that every edge joins a node in A and a node in B.

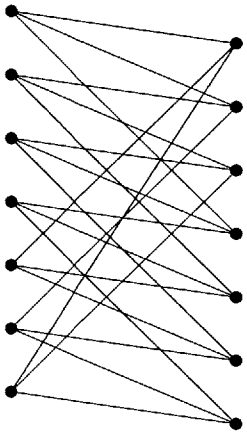


Figure 5.5: *The point-line incidence representation of the Fano plane.*

The point-line incidence graph of a projective plane is a connected, undirected, $n + 1$-regular, bipartite graph. We can now show that the similarity matrix between any pair of graphs, each of which have those four properties, will have identical node and edge similarity matrices.

Let G_A be a r-regular, connected, undirected bipartite graph on n nodes, which each node partition containing $n/2$ nodes. Similarly, let G_B be an s-regular undirected bipartite graph on m nodes, also equally

partitioned. Then without loss of generality, we can assume that the nodes are labeled such that the adjacency matrix of G_A takes the following form:

$$A = \begin{bmatrix} 0 & A_1 \\ A_2 & 0 \end{bmatrix}$$

It is easy to see then, that if $v = \begin{bmatrix} v_1 & v_2 \end{bmatrix}^\top$ is an eigenvector of A with eigenvalue λ, then $w = \begin{bmatrix} v_1 & -v_2 \end{bmatrix}^\top$ is an eigenvector with eigenvalue $-\lambda$. Also, since A has constant row sum r, its Perron root $\rho(A)$ is equal to $\|A\|_\infty = r$, the eigenvector associated with $\rho(A) = r$ is $\begin{bmatrix} 1 & \dots & 1 & 1 & \dots & 1 \end{bmatrix}$ and the eigenvector associated with $-\rho(A) = -r$ is $\begin{bmatrix} 1 & \dots & 1 & -1 & \dots & -1 \end{bmatrix}$

Also, because G_A is connected and undirected, it is irreducible. Perron-Frobenius theory tells us that a nonnegative, irreducible matrix has its Perron root as an eigenvalue with multiplicity 1. Thus, we can conclude that r and $-r$ are eigenvalues of A and each have multiplicity 1. In an analogous way, we can conclude that B has simple eigenvalues s and $-s$.

Now, let us consider the matrix M upon which our update is based. Recall that:

$$M = A \otimes B + A^\top \otimes B^\top + D_A^s \otimes D_B^s + D_A^t \otimes D_B^t$$

For regular, undirected graphs:

$$A = A^\top$$

$$D_A^s = D_A^t = rI_n$$

$$B = B^\top$$

$$D_B^s = D_B^t = sI_m$$

Thus, $M = 2(A \otimes B + rsI_{nm})$. We know that if $Ax = \gamma x$ and $By = \delta y$, then $A \otimes B(x \otimes y) = \gamma\delta(x \otimes y)$. Thus, we can see that the dominant eigenvalue of M will be $2rs$, and will have two associated eigenvectors:

$$\begin{bmatrix} 1 & \dots & 1 & 1 & \dots & 1 \end{bmatrix} \text{ and } \begin{bmatrix} 1 & \dots & 1 & -1 & \dots & -1 \end{bmatrix}.$$

Observe that the subspace spanned by the dominant eigenvectors is *independent* of the exact connectivity structure of G_A and G_B - it is the same subspace for any pair of undirected, regular, bipartite graphs. Thus, every self-similarity and cross-similarity matrix for graphs of projective planes will always be identical, regardless of the initial condition. Also observe that the projection of the all-ones vector onto these dominant eigenvectors will yield the following:

$$\begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix} \begin{bmatrix} 1 & \dots & 1 & 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ \vdots \\ 1 \\ -1 \\ \vdots \\ -1 \end{bmatrix} \begin{bmatrix} 1 & \dots & 1 & 1 & -1 & \dots & -1 \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = 2(n^2 + n + 1) \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

For the all-ones initial condition, every similarity score will be identical.

5.3 Examples

It's worthwhile to compare four different methods of computing node similarity, each with some implicitly associated normalizing step. The first is the method reported in [BVD04], in which $\mathcal{A}$ is the node-node adjacency matrix of graph G_A, and the process is iterated an even number of times:

$$X_{VDB} \leftarrow \mathcal{B}X_{VDB}\mathcal{A}^{\top} + \mathcal{B}^{\top}X_{VDB}\mathcal{A}$$

The second is our coupled node-edge score method, where D_{A_S} is a diagonal matrix containing the out-degree of each node in graph A, and D_{A_I} is a diagonal matrix containing the in-degree of each node in A:

$$X_{NE} \leftarrow \mathcal{B}X_{NE}\mathcal{A}^\top + \mathcal{B}^\top X_{NE}\mathcal{A} + D_{B_S}X_{NE}D_{A_S} + D_{B_I}X_{NE}D_{A_I}$$

The third is a fragment of the modified iteration from [HS03] described above that only includes the novel *positive* terms, so as to maintain a purely positive score:

$$X_{H\hat{S}_+} \leftarrow \mathcal{B}X_{H\hat{S}_+}\mathcal{A}^\top + \mathcal{B}^\top X_{H\hat{S}_+}\mathcal{A} + (I_B - \mathcal{B})X_{H\hat{S}_+}(I_A - \mathcal{A})^\top + (I_B - \mathcal{B})^\top X_{H\hat{S}_+}(I_A - \mathcal{A})$$

The fourth is the entire modified [HS03] update:

$$\begin{aligned} X_{H\hat{S}} \leftarrow\ & \mathcal{B}X_{H\hat{S}}\mathcal{A}^\top + \mathcal{B}^\top X_{H\hat{S}}\mathcal{A} + (I_B - \mathcal{B})X_{H\hat{S}}(I_A - \mathcal{A})^\top + (I_B - \mathcal{B})^\top X_{H\hat{S}}(I_A - \mathcal{A}) \\ & -\mathcal{B}X_{H\hat{S}}(I_A - \mathcal{A})^\top - \mathcal{B}^\top X_{H\hat{S}}(I_A - \mathcal{A}) - (I_B - \mathcal{B})X_{H\hat{S}}\mathcal{A}^\top - (I_B - \mathcal{B})^\top X_{H\hat{S}}\mathcal{A} \end{aligned}$$

We will apply each of these methods to the pair of graphs first presented in Chapter 4 in Figure 4.4 (reprinted here as Figure 5.6, below).

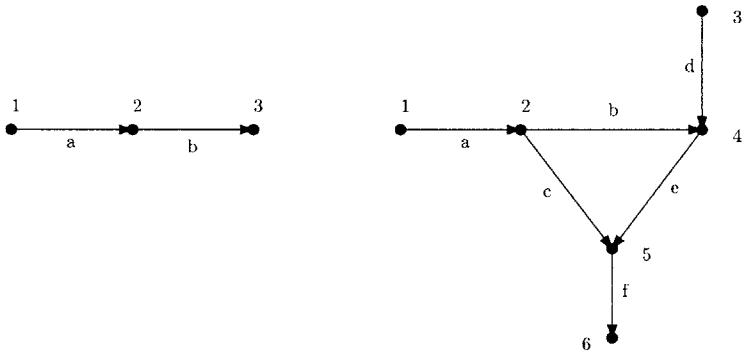


Figure 5.6: *Two directed graphs.*

Intuitively, we might like to see the following correspondences show up in some way:

1. Node $1 \in G_A$ is like node $1 \in G_B$.

2. Node $1 \in G_A$ is like node $3 \in G_B$.

3. Node 2 $\in G_A$ is like node 2 $\in G_B$.

4. Node 2 $\in G_A$ is like node 5 $\in G_B$.

5. Node 3 $\in G_A$ is like node 6 $\in G_B$.

nodes	1	2	3
1	0.211	0	0
2	0.426	0.336	0
3	0.255	0.096	0
4	0.171	0.406	0.271
5	0	0.271	0.466
6	0	0	0.171

Table 5.1: *X_{VDB} for the graphs in Figure 5.6.*

With the [BVD04] score given in Table 5.1, we see that the Correspondences 1 and 3 are represented, but the rest are difficult to distinguish strongly.

nodes	1	2	3
1	0.124	0	0
2	0.348	0.445	0
3	0.157	0.054	0
4	0.094	0.563	0.193
5	0	0.338	0.340
6	0	0	0.094

edges	a	b
a	0.265	0
b	0.426	0.297
c	0.320	0.389
d	0.336	0.115
e	0.202	0.445
f	0	0.202

Table 5.2: *X_{NE} and Y_{NE} for the graphs of Figure 5.6.*

Referring to Table 5.3, we see that zeros appear in the same places for X_{NE} and X_{VDB}, an observation which is true in general. To see this, first recall that the two updates are the same except for the addition of diagonal terms in the coupled model which serve to magnify the scores of the more highly connected nodes. If $[X_{NE}]_{ij} = 0$, then the contribution of these diagonal terms is zero, and the constraints for the [BVD04] update and the coupled model are the same.

Though zeros are located in the same positions, observe that the relative magnitudes of any two scores within each are *not* necessarily the same between the two methods.

For the scores $X_{H\tilde{S}+}$ in Table 5.3, observe that we no longer have zeros in the same locations because of the influence between nodes which are not connected. This method indicates Correspondences 1, 2 and 3, but certainly not 4 or 5.

nodes	1	2	3
1	0.278	0.208	0.262
2	0.237	0.195	0.223
3	0.278	0.210	0.262
4	0.222	0.195	0.239
5	0.223	0.193	0.238
6	0.262	0.209	0.278

Table 5.3: *X_{HS_+} for the graphs in Figure 5.6.*

nodes	1	2	3
1	0.347	0	0.347
2	0.215	0	0.215
3	0.347	0	0.347
4	0.215	0	0.215
5	0.215	0	0.215
6	0.347	0	0.347

Table 5.4: *X_{HS} for the graphs in Figure 5.6.*

These scores given by X_{HS} in Table 5.4 are very difficult to interpret; for example, why is node 2 'unlike' any other node? Or equally 'like' all nodes? One of the difficulties in using positive and negative influences between nodes and edges is that cancellations arise because of symmetries in the graph, yielding scores whose meaning is not easy to decipher.

Chapter 6

Graph matching

ONE COMMON TASK in graph theory applications is the identification of some kind of optimal *matching* between the elements (i.e., nodes and edges) of two graphs. For example, one might use an attributed graph to represent the dominant features of an image of an object, then compare a set of these graphs to determine which depict the same underlying object by aligning the dominant features.

There exist a tremendous number of graph matching algorithms which vary in their specific application, optimization criterion, accuracy, and time and memory complexity. Here, we will restrict our attention to a particular class of techniques that arise in pattern recognition and have a natural connection to our node and edge similarity measures.

6.1 Pattern recognition approaches

Consider the following problem: given two sets of points in a metric space, determine an optimal matching between the elements of the two sets that minimizes some error criterion. Practically, this problem often means matching significant features of two images, when one image is a distortion of the other.

What we seek in this kind of problem is an *assignment matrix* P; if set $\mathcal{A}$ has m elements and set $\mathcal{B}$ 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 one '1' entry in each column. If $P_{ij} = 1$, then element i of $\mathcal{A}$ is matched to element j of $\mathcal{B}$. Note that for $m = n$, P is a permutation matrix.

One early work in point set pattern matching is that of Scott and Longuet-Higgins in [SLH91]. In their formulation, each of the m points in set $\mathcal{A}$ and the n points in set $\mathcal{B}$ has a position in Euclidean space and the distance between each pair of nodes (i, j) for $i \in \mathcal{A}$ and $j \in \mathcal{B}$ is known. These distances are assembled into an $m \times n$ matrix, G, and then each entry is scaled by a Gaussian function to be monotonically decreasing in

distance. Without loss of generality, assume $m \leq n$. In this kind of pattern matching, the desired result is an $m \times n$ assignment matrix, P, that matches the elements in the first image to the elements in the second image so that the sum of the squared Gaussian-scaled distances is maximized; that is, P should be an assignment matrix that maximizes the expression:

$$\sum_i \sum_j P_{ij} G_{ij} = \mathrm{tr}(P^\top G)$$

Instead of explicitly finding this assignment matrix, the algorithm of [SLH91] finds an approximation: the *orthogonal* matrix that maximizes the sum of the squared distances between matched elements (in the case where $m < n$, the term 'orthogonal' is taken to mean that the rows of the matrix are mutually orthogonal). One can then apply a thresholding method to the orthogonal matrix to yield an assignment matrix, but this result is not guaranteed to be optimal.

It is worth observing that this approach is an extension of the work of Umeyama in [U88], in which the author matches two *graphs* with the same number of nodes by finding the orthogonal approximation to a permutation matrix that minimizes the sum of squared differences between the two adjacency matrices. Similarly, Almohamad and Duffuaa present a linear programming formulation in [AD93] that finds a doubly-stochastic matrix P that minimizes the following objective, where $\mathcal{A}$ and $\mathcal{B}$ represent weighted node-node adjacency matrices of two graphs with the same number of vertices:

$$\| \mathcal{A} - P^\top \mathcal{B} P \|_1$$

The *Hungarian algorithm* is then applied to the matrix P to obtain an approximation to the optimal permutation matrix which minimizes this objective. We will return to the Hungarian algorithm in more detail in Section 6.2.

A different approach to point set matching by Shapiro and Brady in [SB92] uses the *intra*-image distance to compare two point sets, rather than the *inter*-image distance. For each of two point sets $\mathcal{A}$ and $\mathcal{B}$, with m and n features, respectively, matrices A and B are constructed which contain the distance between features *within* each point set (analogous to a graph adjacency matrix). The algorithm then computes the eigenvalues

of each matrix, and associates with each feature a point in the vector space of the eigen-basis. If $n > m$, the eigen-basis for point set B is truncated by removing the dimensions corresponding to the $n-m$ smallest eigenvalues. Once each point for each image is represented by a vector in a vector space of the same dimension, a feature similarity matrix is generated, with the ij^{th} entry defined by the Euclidean distance between the i^{th} vector of the first image and the j^{th} vector of the second. With this similarity matrix, a final step similar to that in [SLH91] can be applied to identify a permutation matrix that *minimizes* the sum of squared similarity scores.

6.2 Maximum weight matching in a bipartite graph

In both [SLH91] and [U88], the authors apply a permutation matrix approximation at the last step, then use a thresholding method to obtain a (possibly sub-optimal) matching. This kind of approximation is not strictly necessary, as there exists low time-complexity algorithms for computing the exact maximum weight matching. In graph theory, this problem is known as the *maximum weight bipartite graph matching problem.* We will digress a moment here to discuss this problem, and one algorithmic solution.

A weighted bipartite graph $G(V, E)$ is one in which the nodes of V can be partitioned into two sets A and B such that every edge joins a node in A and a node in B, and each edge e has an associated weight, $w(e)$. A natural example of this type of structure is the relationship between consumers and products, where a weighted edge represents the preference of a particular consumer for a particular product. Often, it is interesting to identify a one-to-one maximum weight matching between the nodes in A and those in B; i.e., select a subset S of the edges in E such that all nodes at either end of edges in S are unique, and such that the sum of the weights of edges of E is maximized. In our consumer-product example, such a matching would correspond to matching unique consumers to unique products so as to maximize total preference. A numerical example is given below in Table 6.2.

One common algorithm for identifying such a maximum weight matching is the *Hungarian algorithm*, described by Kuhn in 1955. The Hungarian algorithm computes in $O(n^3)$ time a maximum weight matching in a bipartite graph where the subsets A and B are of equal size n. Discussion of the details of the algorithm

1	3	**7**
3	2	4
4	**8**	3

Table 6.1: *If the ijth entry represents i's affinity for j, the maximum affinity matching between the rows and columns is given by the bolfaced elements.*

are too lengthy to present here; for an interesting summary of the algorithm, see Frank's short report [F04].[1]

6.3 Using similarity scores for matching

Let us return to the discussion of the point set matching approaches discussed in Section 6.1. Can these same ideas be used to match abstract graphs? The difficulty in applying these approaches is that there is no natural 'distance' measure for abstract graphs; while point sets are typically embedded in a metric space, graphs are not. However, a set of node similarity scores, which incorporates the structural information of both graphs, might provide such a measure. Graph matching algorithms employing this measure may be particularly useful because they allow us to compare two graphs of unequal size, a task for which few satisfactory algorithms exist.

We will devote the rest of this chapter to exploring a simple problem: identifying the location of a subgraph within a larger graph. 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. A set of $m \leq n$ vertices of G_A are selected; the subgraph induced by these m nodes is called G_B. Note that G_B may not be connected. Next, the node similarity matrix for the two graphs is computed, using the coupled model described in Chapter 5. Finally, the Hungarian algorithm is applied to the node similarity matrix to obtain a matching between the nodes of the subgraph and nodes of the original graph that maximizes the sum of squared matched scores. 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 don't correspond to the

[1] Lovász and Plummer's text, [LP86], is an excellent reference for many interesting problems in general graph matching tasks, beyond the weighted bipartite case.

original identification are marked as failures. An example of this procedure, and the correct subgraph node identifications, are given in Figure 6.1.

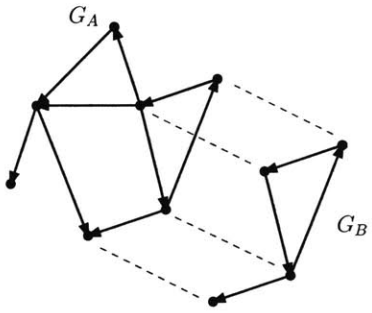


Figure 6.1: *An example of a correct matching between the nodes of a graph G_A and a subgraph G_B, with the correspondences indicated by dashed lines.*

We've applied this procedure to graphs with $n = 15$ and varied the size of the subgraphs and the edge probability (connectivity) parameter p. 500 subgraphs of the specified sizes were generated for each value of p, and the average number of successful matches was computed. The results are given below in Figure 6.2.

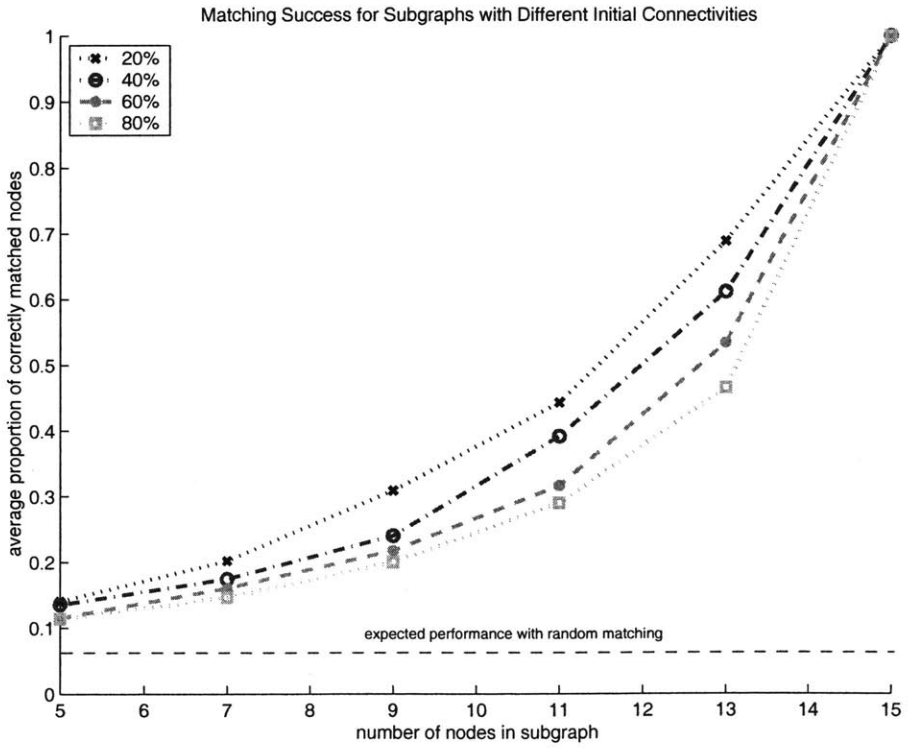


Figure 6.2: *Subgraph matching performance using the Hungarian method applied to the node scores from the coupled model.*

Figure 6.2 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 introductory probability courses. In this problem, n party guests leave their hats in a room; after the party, the hats are randomly redistributed. To twist the problem, imagine that a thief steals $n - m$ of the hats, leaving m remaining. The hats are then distributed randomly to the guests. We would like to know the expected number of correct matches between guest and hat.

Let I_i be the indicator random variable that takes the value 1 when guest i is matched correctly to her hat, and 0 otherwise. Then

$$E[\text{number of correct matches}] = E\left[\sum_{i=1}^{n} I_i\right]$$

The linearity of expectation, the symmetry of the guests, and the nature of indicator random variables allow us to rewrite this expectation as:

$$E\left[\sum_{i=1}^{n} I_i\right] = nP(I_i = 1)$$

Let us consider an intermediate stage in the hat distribution problem, in which m of the n guests are selected to have a hat returned to them. Then we can condition the above probability on two events: that guest i's hat is among the remaining hats, and that guest i is selected to receive a hat. Note that these conditions are independent, and that unless both of the conditions hold, I_i must be 0. Thus,

$$\begin{aligned} nP(I_i = 1) &= nP(I_i = 1 | i \text{ is selected}, i\text{'s hat is among them})P(i \text{ is selected})P(i\text{'s hat is among them}) \\ &= n\left(\frac{(m-1)!}{m!}\right)\left(\frac{m}{n}\right)\left(\frac{m}{n}\right) = \frac{m}{n} \end{aligned}$$

Thus, our expected *proportion* of correct matches for a subgraph of size m is $1/m$, the constant function of subgraph size plotted in Figure 6.2. It is encouraging that our approach is a substantial improvement over the random case! Additionally, we can observe some interesting features of the data of Figure 6.2. First, we see that higher initial connectivities imply lower performance. This is intuitively reasonable; with more

edges, the probability that two nodes share characteristics (i.e., having both in- and out-edges) increases. Also observe that the larger the subgraph, the better our ability to match its nodes correctly. Finally, note that when the subgraph is the *entire* graph, our procedure performs the matching with 100% accuracy every time. We pursue this observation further in Section 6.5.

Although we will focus on maximum weight matchings, it is possible to use the node similarity matrix to obtain a matching using other criteria. In [MGMR02], the authors define their own node similarity measure, then apply 'filters' to the result to obtain a desired many-to-one or one-to-one matching based on specific problem heuristics; among these filters are the *stable marriage* matching criterion and the *perfectionist egalitarian polygamy* criterion.

Before we continue, we should note that Heymans and Singh in [HS03] generated a maximum weight bipartite matching on similarity scores as an intermediate step before generating a single matching 'score' for the similarity of two metabolic networks (see Chapter 4 for a discussion of the [HS03] scoring system). We will discuss related work on performing matching in protein interaction networks in Appendix A.

6.4 Improving matching performance

It should not be surprising that the performance results presented in the previous section are not exceptionally good, especially for the smaller subgraphs. Algorithms that are designed to give good matching performance incorporate many heuristics, are typically application-specific, and are computationally intensive. We should not, therefore, expect our simple iterative computation, followed by an application of the $O(n^3)$ Hungarian algorithm, to outperform these more directed approaches. However, it is worth considering what other pieces of information we might be able to use to improve the performance of this approach, given the baseline of Figure 6.2. Here, we discuss several of these improvements.

6.4.1 Penalizing for degree mismatch

If we know *a priori* that the graph G_B is a subgraph of G_A, then the in- and out-degree of node a_i of G_A must be greater than or equal to the in- and out-degrees of its corresponding node b_i of G_B. We can incorporate this extra information by setting the similarity score of two nodes which violate this relationship to a large

negative number. The results of this modified approach are given in Figure 6.3, where the baseline and penalized performance are both represented. The data in this figure indicates no perceptible performance change when penalties are used. Apparently, the node candidates that are removed via penalties are not the incorrect matches identified by the unpenalized maximum-weight matching.

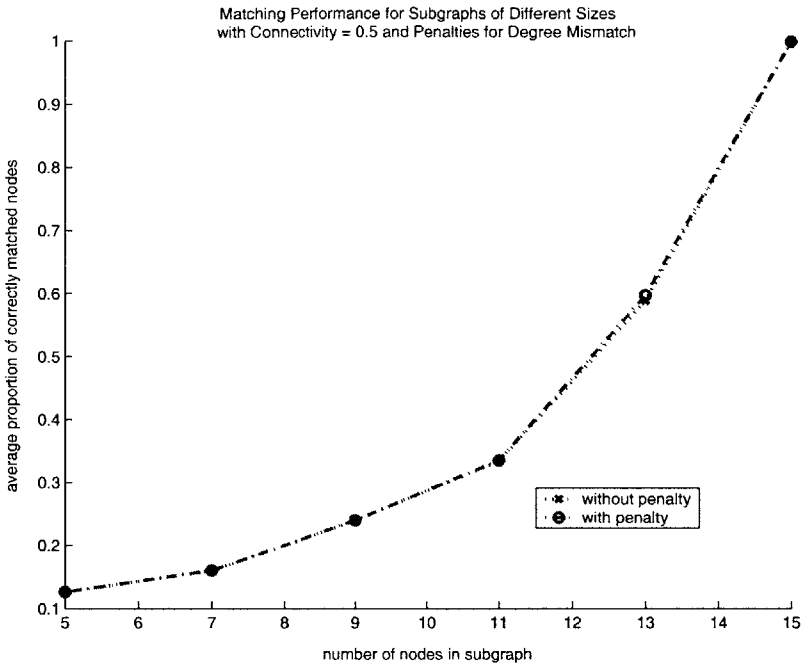


Figure 6.3: *Performance after including a penalty for degree mismatch of subgraphs.*

6.4.2 Incorporating local edge similarity scores

Observe that our initial approach to matching did not *explicitly* use the edge similarity scores; since the node and edge scores are related by a simple equation, it is not apparent that edge scores can provide us with any additional information. With a little bit of extra work, however, we might be able to use the edge scores in a constructive way.

First, recall the definition of the similarity score between two nodes, i and j; x_{ij} is simply the sum of the edge scores of the edges incident to each node. Observe that this simple definition does not reflect the 'macro' similarity of each neighborhood. To restate this, the score x_{ij} is the sum of similarities between adjacent edges considered pairwise, not a measurement of the *global similarity* of the neighborhoods.

In matching, we might want to take into account an extra degree of local similarity. Consider the following idea. First, compute the node-node similarity score, X_{ij}, for nodes i of G_B and node j of G_A. Next, look at $O(i)$, the set of all outgoing edges of i, and $O(j)$, the set of all outgoing edges of j. Define $I(i)$ and $I(j)$ analogously for incoming edges. Next, apply the Hungarian algorithm to the submatrix of the set of edge-similarity scores, Y, which correspond to the edges in $O(i)$ and $O(j)$, and compute the weight of the maximum-weight edge matching, $O_{\max}$. Repeat this process with the submatrix of Y corresponding to the sets of edges $I(i)$ and $I(j)$, yielding a matching weight $I_{\max}$. Finally, update the node-node similarity score X_{ij} by:

$$X_{ij} \leftarrow X_{ij} + \gamma(O_{\max} + I_{\max})$$

where γ is referred to as the *edge match multiplier*. The Hungarian method can now be applied to this updated matrix X to determine a node-matching.

Some performance results of this procedure are given below in Figure 6.4. When $\gamma = 0$, the matching is performed on the original X matrix. Interestingly, there appears to be an initial improvement in performance when γ is greater than zero, but that effect levels out as γ is increased further.

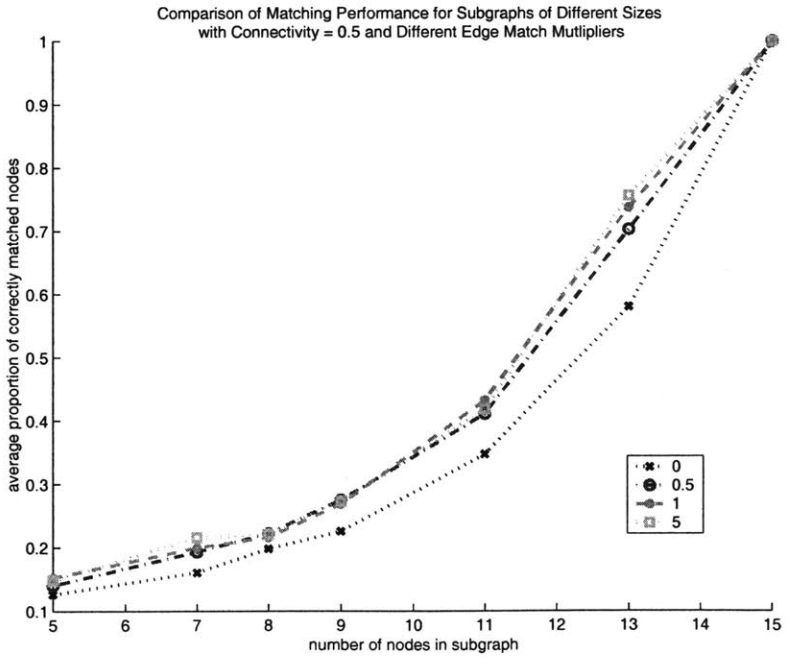


Figure 6.4: *Performance after augmenting the node scores via local matching on the edges.*

It is also worth considering the effect of iterating this process, i.e. updating the node similarity score matrix X as described above, then recomputing the edge similarity scores Y as follows:

$$Y \leftarrow B_S^T X A_S + B_T^T X A_T$$

The node similarity scores X can be updated again using the Hungarian matching on local edges via the new edge scores Y. Performance results for a repeated application of this approach are given in Figure 6.5. In this figure, we have plotted a histogram of the number of correct node matches out of 500 trials for subgraphs of different sizes. The average number of correctly matched nodes for each number of applications of the 'local Hungarian' and each subgraph size are drawn as vertical lines.

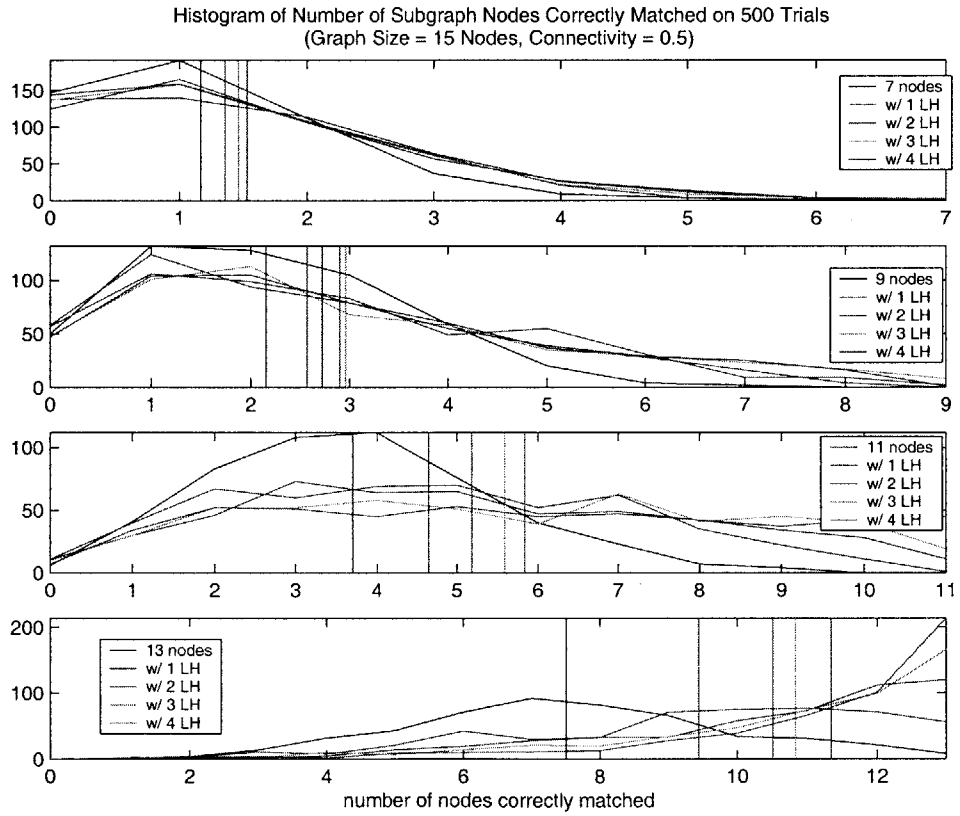


Figure 6.5: *Repeated application of the 'local Hungarians' on the edge scores.*

In Figure 6.5, we see that the matching performance improves with each application of the node score update procedure. These performance improvements are strongest for the larger subgraphs. For the 13 node

subgraph, we also see a change in the *shape* of the histogram after two applications of the 'local Hungarian' procedure; the mode of the distribution occurs at 15 nodes correctly matched.

6.4.3 Employing node labels

Often in graph matching applications, there exists side information about the attributes of nodes and edges. For example, in a protein interaction network, each node represents a unique and known protein. The node matching procedures that we've discussed thus far have used only graph topology to identify similarities; including node label information to penalize or reward certain matches may improve performance.

To test the impact of this kind of information, we slightly modify our previous subgraph matching procedure. A set of s node labels is generated and assigned with equal probability to each node in the graph 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 6.6. Here, we have plotted a histogram 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.

We might also ask what happens when there are only two node labels, which are distributed with asymmetric probabilities. In Figure 6.7, we present a histogram 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 and the better the performance.

6.4.4 Using unique labels

In some applications, we may have knowledge about the correct correspondence between two nodes. For example, if two graphs represent two different images of an object, perhaps there exists a reference node in each graph that is known to correspond to a specific point on the object. We can then match these two nodes with confidence. It is interesting to consider how matching performance changes as known correspondences are taken into account.

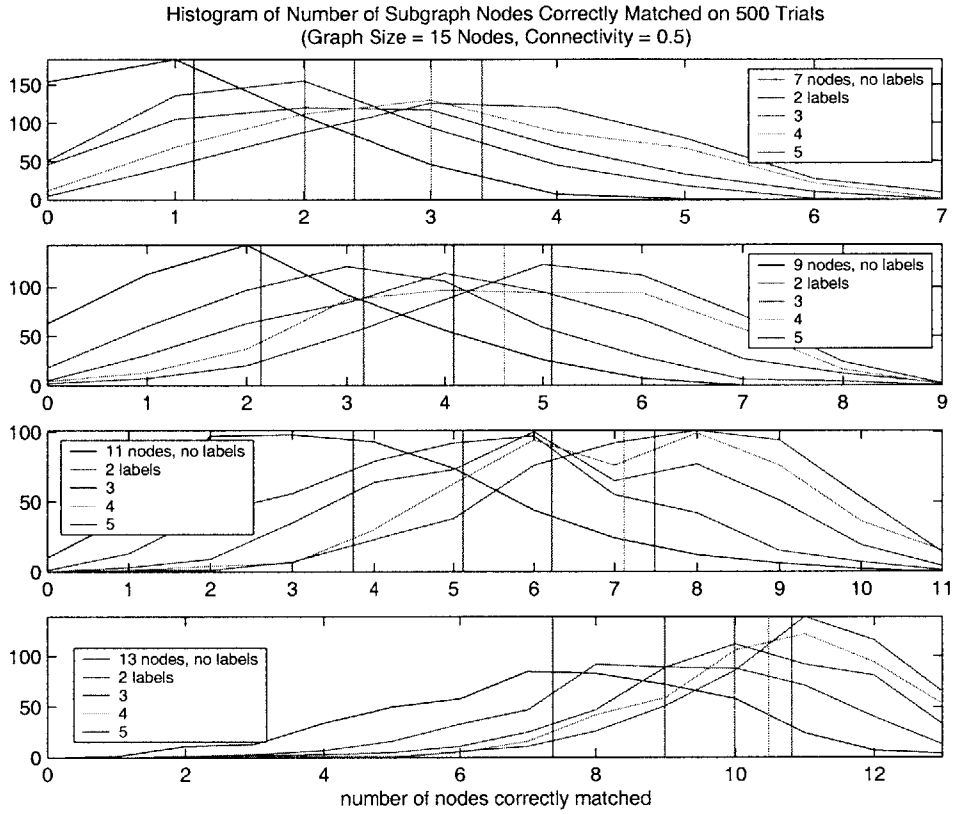


Figure 6.6: *Matching performance using 0-5 node labels.*

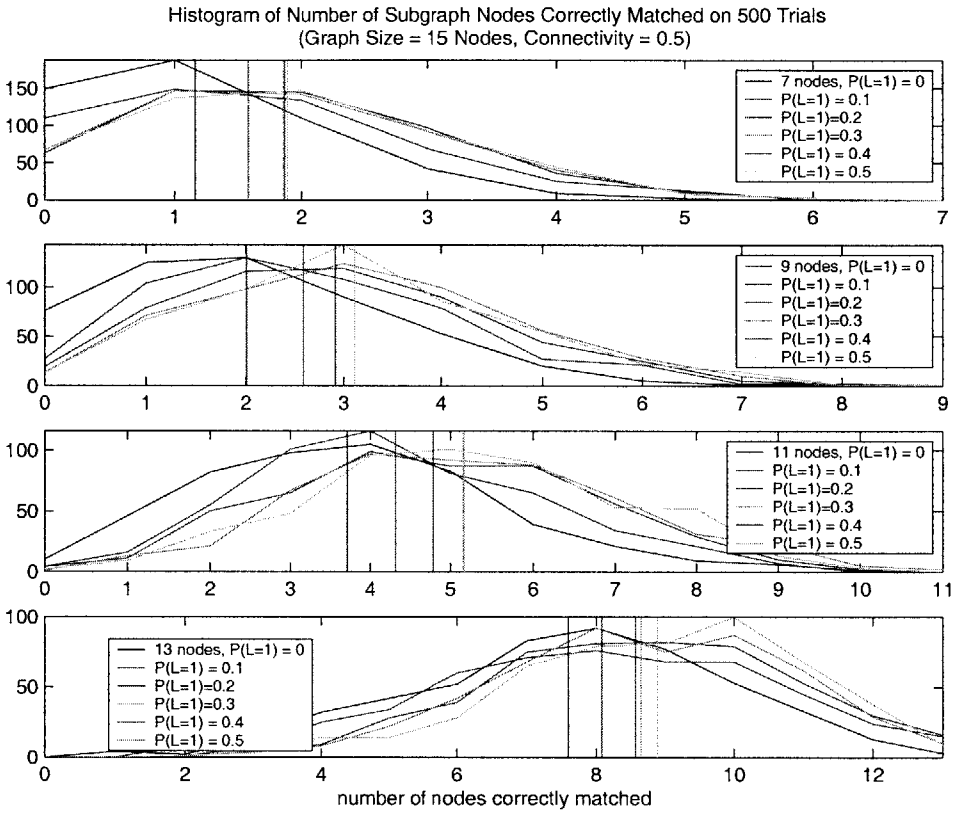


Figure 6.7: *Matching performance using 2 node labels distributed with different probabilities.*

To test this idea with our subgraph matching framework, we first generate a random graph G_A with n nodes, then label s of its nodes with unique identifiers and label the remaining $n - s$ with the same label. A random subgraph G_B is removed, carrying the node labels along. We then compute the node similarity matrix between G_A and G_B, then penalize the similarity scores with mismatched labels by setting it to a large negative number. The Hungarian algorithm is applied to yield a maximum weight matching, and the average proportion of correctly matched nodes is computed.

In Figure 6.8, each curve represents the performance for a particular value of s as it is varied from 0 to 14 on a 15 node graph.

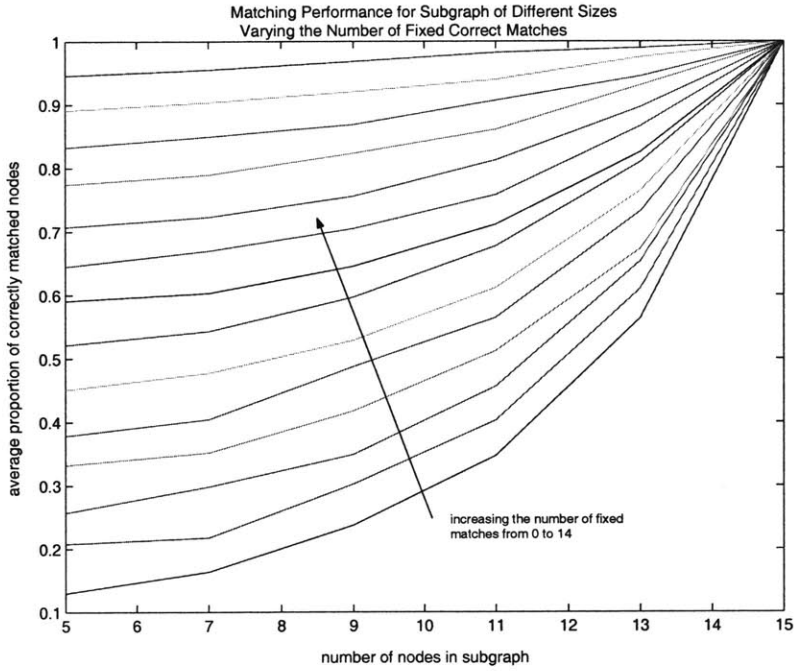


Figure 6.8: *Matching performance using varying numbers of unique labels.*

These results imply a fixed improvement in performance for every additional unique label.

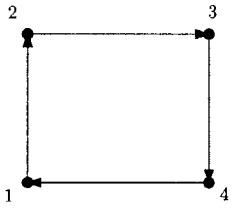
6.5 Identifying isomorphisms

The previous sections have focused on using node similarity scores to correctly position a subgraph within a larger graph. However, it is also useful to be able to identify a correspondence between two graphs of the

same size that are potentially isomorphic.

Consider the following test. First, a random graph G_A with n nodes and connectivity p is generated, with adjacency matrix $\mathcal{A}$. Next, a random permutation matrix P is generated, and a new adjacency matrix $\mathcal{B} = P^{\top}\mathcal{A}P$ is computed. The node similarity matrix, X, of G_A and G_B is calculated, and the Hungarian algorithm is applied to X to find the maximum weight matching. This matching can be represented by an assignment matrix, C. If $C = P$, then the algorithm has correctly identified an isomorphism between G_A and G_B (there may exist other isomorphisms, but C is certainly one).

This simulation was run for many different choices of n and p and $C = P$ in every trial. Unfortunately, this naive approach will not always work for graphs with a high degree of symmetry; for example, the cycle graph with four nodes. This graph and its self-similarity matrix are shown below in Figure 6.9. The self-similarity matrix of this graph has all entries the same, so *any* matching is a maximum weight matching, although many of these matchings are not isomorphisms. At the same time, it is true that a correct isomorphism is among the set of *all* maximum weight matchings.



nodes	1	2	3	4
1	0.250	0.250	0.250	0.250
2	0.250	0.250	0.250	0.250
3	0.250	0.250	0.250	0.250
4	0.250	0.250	0.250	0.250

Figure 6.9: *The 4-cycle graph and its self-similarity score matrix.*

This leads us to a conjecture. Let G_A and G_B be isomorphic graphs. Without loss of generality, we can assume that their adjacency matrices are the same. If this is not true, then their adjacency matrices are related by conjugation with a permutation matrix, which yields a conjugated similarity matrix. Denote the similarity matrix by X. An isomorphic assignment between the nodes of G_A and G_B can be represented with the identity matrix. Again, observe that the identity mapping may not be the *only* isomorphism between G_A and G_B, but it is certainly one of them. The weight of this particular matching will be $\mathrm{tr}(X)$. The weight of any other matching is given by $\mathrm{tr}(PX)$ where P is the permutation matrix which represents the matching.

What we would like to show, or find a counterexample to, is the following: for any permutation matrix P,

$$\mathrm{tr}(X) \geq \mathrm{tr}(PX)$$

where X satisfies the following equation

$$\gamma X = \mathcal{A}X\mathcal{A}^{\top} + \mathcal{A}^{\top}X\mathcal{A} + D_A^t X D_A^t + D_A^s X D_A^s$$

for $\mathcal{A}$ a fixed node-node adjacency matrix, D_A^t a diagonal matrix containing the column sums of $\mathcal{A}$, D_A^s a diagonal matrix containing the row sums of $\mathcal{A}$, and γ a fixed constant required by the normalization (in fact, γ is the dominant eigenvalue of the matrix $M = \mathcal{A} \otimes \mathcal{B} + \mathcal{A}^{\top} \otimes \mathcal{B}^{\top} + D_{A_S} \otimes D_{B_S} + D_{A_T} \otimes D_{B_T}$).

Observe that $\mathrm{tr}(PX)$ is equal to $< \mathrm{vec}(P^{\top}), \mathrm{vec}(X) >$. If we represent the entire set of (appropriately-sized) permutation matrices as $\mathcal{P}$, then we can write this question as an optimization problem as follows:

$$\begin{aligned} \max_P \quad & < \mathrm{vec}(P^{\top}), \mathrm{vec}(X) > \\ \text{s.t.} \quad & (\gamma I - M)\mathrm{vec}(X) = 0 \\ & P \in \mathcal{P} \end{aligned}$$

Proving the conjecture is equivalent to showing that $P = I$ is an optimal solution to this problem.

Chapter 7

Conclusions and ongoing work

MANY EXTENSIONS and suggestions have been proposed throughout this thesis; here, we'll propose a few ideas for future work.

- **Similarity scores as structural markers and isomorphism detection**

  The connectivity structure of the graphs G_A and G_B dictates the similarity score matrices. What kinds of structural features of G_A or G_B that can be inferred from patterns in the similarity matrices? In [BVD04], the authors compute a closed-form solution for their node similarity measure between the path graph on three nodes and an arbitrary graph. They also discuss some conditions under which their node similarity matrix will have rank 1. In Section 5.2, we have explored the form of self-similarity matrices and the node similarity matrices of projective planes. As discussed in Section 5.2.5, one way of visualizing these methods of calculating similarity is to picture similarity scores as flows on the *product graph* of G_A and G_B. It should be possible to obtain additional structural results using this connection between flow problems and product graph topology.

  We have conjectured in Section 6.5 that if two graphs are isomorphic, then an isomorphic mapping between their nodes is among the maximum weight matches. Regardless of whether this conjecture is true or false, the self-similarity matrices of some graphs communicate very little or no interesting information; see, for example, the bipartite graphs that represent projective planes discussed in Section 5.2.8, whose self-similarity scores are uniform between every node pair. While projections onto the dominant eigenspace associated with the matrix M 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 (and can be explored by looking at the rate of convergence of the iteration, for example). For many kinds of graphs, the spectrum of the node-node adjacency matrix is well-characterized (see,

for example, [CDS80]); using the machinery of spectral graph theory to explore the spectrum of the matrix M would be very worthwhile. As a first step in this direction, consider the *signless Laplacian matrix*, $|L|$, defined in [vDH03] as:

$$|L| = \mathcal{D}_{\mathcal{A}} + \mathcal{A}$$

where $\mathcal{A}$ is the node-node adjacency matrix of an undirected graph G_A, and $\mathcal{D}_{\mathcal{A}}$ is a diagonal matrix with the degrees of the nodes of G_A on the diagonal. Recall the node similarity score update procedure from Section 5.2.4 :

$$\begin{aligned} x_k &\leftarrow \left(\mathcal{A} \otimes \mathcal{B} + \mathcal{A}^{\mathsf{T}} \otimes \mathcal{B}^{\mathsf{T}} + D_{A_S} \otimes D_{B_S} + D_{A_T} \otimes D_{B_T}\right) x_{k-1} \\ &\equiv \left(P_{\mathcal{AB}} + \mathcal{D}_{\mathcal{AB}}\right) x_{k-1} \end{aligned}$$

Here, $P_{\mathcal{AB}}$ is the node-node adjacency matrix of the product graph of G_A and G_B and $\mathcal{D}_{\mathcal{AB}}$ is diagonal with the degrees of the nodes of the product graph G_{AB}. Thus, the node similarity scores between two graphs G_A and G_B are a projection onto the invariant subspace associated with the Perron root of the signless Laplacian of their product graph G_{AB}. The signless Laplacian, and its better known counterpart, the Laplacian $L = \mathcal{D}_{\mathcal{A}} - \mathcal{A}$, are the central objects of study in many branches of spectral graph theory.

- **Utilizing edge scores and considering hypergraphs**

  As discussed in Chapter 4, the previous work with iterative similarity methods focuses exclusively on *node* scoring, largely because it is often the more intuitive object; many real world graphs place the elements of interest at the nodes, and are not as concerned with the connections between them. We have exhibited this same bias here - in our discussion of matching, we have only utilized the edge scores in one of our performance-improving heuristics. Edges are not always of secondary interest! Indeed, edges are often of principal interest, particularly when a *hypergraph* structure applies to the

data. Hypergraphs, as discussed briefly in Chapter 2, arise when one is interested in modeling relationships between *groups* of objects. One example of a hypergraph structure is the description of chemical reactions within a cell; a group of molecules is connected with a hyperedge if they all co-participate in a particular reaction. When comparing two graphs of this type, it is the edges, or reactions, that might be most interesting to examine.

Iterative types of similarity methods have not yet been applied to hypergraphs, but the extension is not difficult to imagine: a node i in hypergraph G_A is similar to a node in hypergraph G_B proportionally to the similarity of the edges to which i and j belong. Analogously, two hyperedges are similar as the sum of the similarity of the nodes that they contain. A challenge lies in interpreting these scores, and finding appropriate applications for this kind of measure.

- **Applications**

One interesting application of graph similarity methods is the identification of functionally similar subgraphs within maps of protein interactions within different species. Appendix A discusses the task of subgraph matching within the yeast interactome. Both node information and matching heuristics have been applied to this task and have improved matching performance. Using biological network data to derive meaningful observations is a very active area of research; as one example, Berg and Lässig developed a probabilistic model of biological network evolution to locate significant repeated subgraphs which they call *motifs* [BL04]. The omnipresence of natural phenomena with a graph structure has inspired many different applications: Kleinberg's motivation in [K99] was an improved web search tool, while Blondel et. al [BVD04] applied their method to automatic synonym extraction from a dictionary graph. Several other applications for iterative similarity methods were discussed in Chapter 4. Finding additional appropriate applications for this kind of matching might inspire some interesting new heuristics, or an entirely new approach.

Appendix A

Protein interaction networks

As a possible application of the graph matching routines that were discussed in Chapter 6, we have explored some of the publicly available protein interaction data. Although these explorations were quite preliminary, they certainly informed our perspective on graph-matching issues and are thus worth documenting.

One interesting problem in bioinformatics is the identification of orthologous processes between species; that is, given a specific biological process in one species, we would like to identify the analogous process in another species. Often, the processes of interest are involved in *metabolism*, the collection of all internal reactions that allow a cell to maintain itself, grow, and eventually die. There are several species that have traditionally been considered 'model organisms' whose metabolic processes are relatively well-annotated: the worm *C. elegans*, the fruitfly and the mouse are among them.

A.1 The *S. cerevisiae* interactome

We have begun by focusing on another of these model organisms, *S. cerevisiae*, a budding yeast, and examining its protein-protein interaction graph or *interactome*. Our source of data on these interactions is the Database of Interacting Proteins^{TM}(DIP), a comprehensive source of protein interaction data culled from several other databases and the biological literature [DIP05]. The yeast interactome is constructed by nodes which represent unique proteins found in yeast cells; two nodes are connected if they have been found to bind *in vitro*. Observe that this definition leads to a graph whose edges are undirected. The DIP contains two sets of yeast interaction data - a *full* set of all documented interactions, and a *core* set containing interactions that have been more reliably validated. The full interactome contains 4741 different proteins, while the core interactome contains 2635 unique proteins. There are 6574 edges in the core interactome, corresponding to an average connectivity of 0.0019. There are 92 disconnected components in the core interactome (the number

of eigenvalues at zero of the Laplacian of the graph), and each protein is coded by a unique DIP number. Our work with the yeast interactome is restricted to this core set.

We would like to be able to identify biologically relevent subgraphs of the interactome; eventually, these subgraphs could be compared to the interactomes of other species to identify orthologous processes. To obtain these subgraphs, we use the Comprehensive Yeast Genome Database (CYGD), sponsored by the Munich Information Center for Protein Sequences [CYGD05]. The CYGD catalogs proteins in many ways, one of which is by participation in known *complexes*. We considered several complexes which correspond to subgraphs of the interactome, including the cytoskeleton (CYT) complex with 73 proteins and the intracellular transport (TPORT) complex with 95 proteins. Each of these complexes is specified as a list of proteins. The subgraph of the interactome associated with each complex is obtained by identifying the nodes in the interactome corresponding to the proteins in the complex list.

There exist a number of pragmatic difficulties that arise when combining data from several different databases. While the proteins from the DIP core interactome are identified by their DIP number, the proteins in the CYGD are labeled by their Open Reading Frame (ORF) number. In order to translate between the two labels, we can use the DIP full interactome as a key, which includes both the DIP and ORF labels. Interestingly, there exist seven proteins in the core interactome which are *not* contained in the full interactome - these are removed from our core graph. Additionally, there are proteins in the CYGD complex lists which are not represented in the full interactome - seven from the CYT complex and four from the TPORT complex. These were eliminated from the complex lists.

A summary of the number of proteins and interactions in each of the graphs of interest is given in Table A.1.

	Initial Graphs		Post-Editing	
	proteins	interactions	proteins	interactions
CORE	2635	6574	2628	6556
CYT	73	–	66	185
TPORT	95	–	91	406

Table A.1: *Some properties of the yeast interactome and subgraphs.*

A.2 Identifying subgraphs in the interactome

As discussed early in this Appendix, one is often interested in comparing orthologous processes between different species. Insofar as interesting features of these processes are reflected in the topology of the protein-protein interaction networks of each species, comparing the structure of these different networks might reveal some patterns with biological significance. One way of performing this kind of cross-species comparison is to perform a matching between the interactome of species A and the sub-interactome of species B which corresponds to a specific function. If this matching identifies a sub-interactome of A that closely aligns with that of B, we might look for a similar function in species A.

As a first step toward this type of matching, it is worth looking at the performance of a simple subgraph matching procedure as discussed in Chapter 6. The subgraphs we choose are the TPORT and CYT subgraphs discussed in the previous section, which correspond to complexes within the yeast cell. Some performance results of this procedure are given in Table A.2.

	no. matched correctly	max. weight	weight of correct match	weight of random match (std. dev)
CYT	0	1.0086	4.48e-4	4.17e-4 (4.19e-4)
TPORT	0	0.7225	2.08e-5	6.13e-4 (8.04e-4)

Table A.2: *Performance results of subgraph matching in the yeast interactome.*

It is not surprising that topology alone is not enough to correctly position these subgraphs within the interactome; the size of the subgraphs is less than 4% of the size of the entire interactome, so we should expect performance worse than the 5-node subgraph of the randomly generated 15-node graphs considered in Chapter 6. It is worthwhile, then, to see if some of the heuristics developed in Chapter 6 will improve performance here. Since we have the DIP identifications of each node available, one natural augmentation is the inclusion of node labels. As a first step, we correctly matched a number s of the nodes *a priori* via their node labels (with the correctly matched nodes randomly selected), then examined whether this information

allowed additional nodes to be matched correctly; we found no improvement in matching under this approach. A more fruitful second approach constrained the *neighborhoods* of the correctly matched nodes to be matched to each other. The results from three trials on each of the complex subgraphs is given below in Table A.3.

	number of a priori correct matches	average final number of correct matches
CYT	10	15.2
	20	33.0
	30	48.8
	40	58.6
	50	63.6
	60	65.4
TPORT	10	21.3
	20	46.0
	30	72.3
	40	84.7
	50	88.0
	60	91.0
	70	91.0
	80	91.0

Table A.3: *Performance results with a given number of correct matchings and constrained neighborhoods.*

It is clear that both node information and local edge information are necessary when matching these small complex subgraphs within the larger interactome; future work should explore additional methods for exploiting this information.

Bibliography

[AA03] Adamic, L., Adar, E. (2003). Friends and neighbors on the Web. *Social Networks*, v. 25, 211-230.

[AA04] Adamic, L., Adar, E. (2004). How to search a social network. *HP Labs*, Palo Alto, CA.

[AB02] Albert, R., Barabasi, A.L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, v. 74, 47-97.

[AD93] 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.

[B73] Berge, C. (1973). *Graphs and hypergraphs*. North-Holland Publishing Co., Paris.

[B85] Balaban, A.T. (1985). Applications of graph theory in chemistry. *Journal of Chemical Information and Computer Sciences*, v. 25, 334-343.

[B99] 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.

[BBC04] Beard, D.A., Babson, E., Curtis, E., Qian, H. (2004). Thermodynamic constraints for biochemical networks. *Journal of Theoretical Biology*, v. 228(3), 327-333.

[BH90] Buckley, F., Harary, F. (1990). *Distance in Graphs*, Addison-Wesley.

[BL04] Berg, J., Lässig, M. (2004). Local graph alignment and motif search in biological networks. *Proceedings of the National Academy of Science*, v. 101(41), 14689-14694.

[BVD04] 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.

[BNVD04.2] Blondel, V., Ninove, L., Van Dooren, P. (2004). Affine iterations on nonnegative vectors. *MTNS Symposium 2004*.

[BP98] Brin, S., Page, L. (1998). Anatomy of a large-scale hypertextual web search engine. *7th International World Wide Web Conference*, Brisbane, Australia; 107-117.

[BJK00] Bunke, H., Jiang, X., Kandel, A. (2000). On the minimum common supergraph of two graphs. *Computing*, v. 65, 13-25.

[BS98] Bunke, H., Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, v. 19, 255-259.

[C97] Chugn, F.R.K. (1997). Spectral Graph Theory. American Mathematical Society.

[CK04] Caelli, T., Kosinov, S. (2004). An eigenspace projection clustering method for inexact graph matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, v. 26, 515-519.

[CH03] Carcassoni, M., Hancock, E.R. (2003). Correspondence matching with modal clusters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 25(12), 1609-1615.

[CLDG03] Chung, F., Lu, L.Y., Dewey, T.G., Galas, D.J. (2003). Duplication models for biological networks. *Journal of Computational Biology*, v. 10(5), 677-687.

[CFSV04] Conte, D., Foggia, P., Sansone, C., Vento, M. (2004). Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, v. 18(3), 265-298.

[CG70] Corneil, D.G., Gotlieb, C.C. (1970). An efficient algorithm for graph isomorphism. *Journal of the ACM*, v. 17(1), 51-64.

[CCMZ03] Cristo, M., Calado, P., de Moura, E.S., Ziviani, N., Ribeiro-Neto, B. (2003). Link information as a similarity measure in web classification. In *Proceedings of String Processing and Information Retrieval*, Lecture Notes in Computer Science, v. 2857, 43-55.

[CYGD05] Güldener, U., Münsterkötter, M., Kastenmüller, G., Strack, N., van Helden, J., Lemer, C., Richelles, J., Wodak, S.J., Garcia-Martinez, J., Perez-Ortin, J.E., Michael, H., Kaps, A., Talla, E., Dujon, B., Andre, B., Souciet, J.L., De Montigny, J., Bon, E., Gaillardin, C., Mewes, H.W. (2005) CYGD: the Comprehensive Yeast Genome Database. *Nucleic Acids Research*, v. 33 (D364-8).

[CDS80] Cvetković, D., Doob, M., Sachs, H. (1980). *Spectral of graphs: theory and application*. Academic Press.

[DIP05] Database of Interacting Proteins™. Copyright 1999-2004 UCLA. `http://dip.doe-mbi.ucla.edu/dip/`. Last accessed: 03/16/2005.

[vDH03] van Dam, E.R., Haemers, W.H. (2003). Which graphs are determined by their spectrum? *Linear Algebra and its Applications*, v. 373, 241-272.

[DYB03] Davis, G.F., Yoo, M., Baker, W.E. (2003). The small world of the American corporate elite, 1982-2001. *Strategic Organization*, v. 1(3), 301-326.

[DKMR02] Dill, S., Kumar, R., McCurley, K., Rajagopalan, S., Sivakumar, D., Tomkins, A. (2002). Self-similarity in the web. *ACM Transactions on Internet Technology*, v. 2(3), 205-223.

[DSDK04] Demirci, M., Shokoufandeh, A., Dickinson, S., Keselman, Y., Bretzner, L. (2004). Many-to-many feature matching using spherical coding of directed graphs. *Computer Vision - ECCV 2004*, Lecture Notes in Computer Science, v. 3021, 322-335.

[DZHH04] Ding, C., Zha, H., He, X., Husbands, P., Simon, H. (2004). Link analysis: hubs and authorities on the World Wide Web. *SIAM Review*, v. 46(2), 256-268.

[F04] Frank, András. (2004). On Kuhn's Hungarian method - a tribute from Hungary. *Egerváry Research Group on Combinatorial Optimization Technical Report*, TR-2004-14. `http://www.cs.elte.hu/egres/tr/egres-04-14.pdf`.

[FLF94] Fang, Y., Loparo, K., Feng, X. (1994). Inequalities for the trace of matrix product. *IEEE Transactions on Automatic Control*, v. 29(12), 2489-2490.

[FV01] 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.

[G04] Google Blog - Bill Coughran. (2004). Google's index nearly doubles. `http://www.google.com/googleblog/2004/11/googles-index-nearly-doubles.html`. Last accessed: 03/08/2005.

[GJ79] Garey, M.R., Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.

[H00.1] Hayes, B. (2000). Graph theory in practice: Part I. *American Scientist*, v. 88(1), 9-13.

[H00.2] Hayes, B. (2000). Graph theory in practice: Part II. *American Scientist*, v. 88(2), 104-109.

[HJ85] Horn, R., Johnson, C. (1985). *Matrix Analysis*. Cambridge University Press.

[HOGK03] Hattori, M., Okuno, Y., Goto, S., Kanehisa, M. (2003). Development of a chemical structure comparison method for integrated analyses of chemical and genomic information in the metabolic pathways. *Journal of the American Chemical Society*, v. 125, 11853-11865.

[HS03] Heymans, M., Singh, A. (2003). Deriving phylogenetic trees from the similarity analysis of metabolic pathways. *Bioinformatics*, v. 19(Supp.1), 138-146.

[JK01] Jain, S. Krishna, S. (2001). A model for the emergence of cooperation, interdependence and structure in evolving networks. *Proceedings of the National Academ of Sciences of the United States of America*, v. 98(2), 543-547.

[JW02] 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.

[JMB01] Jiang, X.Y., Munger, A., Bunke, H. (2001). On median graphs: properties, algorithms and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 23(10), 1144-1151.

[KB03] Kanehisa, M., Bork, P. (2003). Bioinformatics in the post-sequence era. *Nature - Genetics*, v. 33, 305-310.

[KKT03] Kempe, D., Kleinberg, J.M., Tardos, E. (2003). Maximizing the spread of influence through a social network. *Proc. 9th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining.*

[K99] Kleinberg, J.M. (1999). Authoritative souces in a hyperlinked environment. *Journal of the ACM*, v. 46, 614-632.

[L91] Lam, C. The search for a finite projective plane of order 10. *The American Mathematical Monthly*, v. 98(4), 305-318.

[L71] Levi, G. (1971). A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo*, v. 9, 341-354.

[L93] Lovàsz, L. (1993). Random walks on graphs: a survey. *Bolyai Society Mathematical Studies 2*, 1-46.

[LP86] Lovàsz, L., Plummer, M. (1986). *Matching Theory*. Elsevier Publishing, NY.

[M81] McKay, Brendan. (1981). Practical graph isomorphism. *Congressus Numerantium*, v. 30, 45-87. `http://cs.anu.edu.au/~bdm/nauty/`.

[M97] Mohar, B. (1997) Some applications of Laplace eigenvalues of graphs. *Graph Symmetry: Algebraic Methods and Applications*, 497 NATO ASI Series C, 227-275.

[MB98] Messmer, B.T., Bunke, H. (1998). A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, 493-504.

[MGMR02] 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.

[P99] Pelillo, M. (1999). Replicator equations, maximum cliques and graph isomorphism. *Neural Computation*, v. 11(8), 1933-1955.

[P02] Pelillo, M. (2002). Matching free trees, maximal cliques and monotone game dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, 1535-1541.

[PSZ99] Pelillo, M., Siddiqi, K., Zucker, S. (1999). Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 21, 1105-1119.

[PRPF03] Price, N.D., Reed, J.L., Papin, J.A., Famili, I., Palsson, B.O. (2003). Analysis of metabolic capabilities using singular value decomposition of extreme pathway matrices. *Biophysical Journal*, v. 84(2), 794-804.

[QH04] Qiu, H.J., Hancock, E.R. (2004). Spectral simplification of graphs. *Computer Vision - ECCV 2004, Pt. 4*, v. 3024, 114-126.

[R04] Robinson, Sara. (2004). The ongoing search for efficient web search algorithms. *SIAM News*, v. 37(9).

[R05] Royle, G. Gordon Royle's Planes of Order 16. `http://www.csse.uwa.edu.au/~gordon/remote/planes16/index.html`. Last accessed: 03/08/2005.

[RM96] Rangarajan, A., Mjolsness, E. (1996). A Lagrangian relaxation network for graph matching. *IEEE Transactions on Neural Networks*, v. 7(6), 1365-1381.

[RW02] Raymond, J.W., Willett, P. (2002). Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, v. 16(7), 521-533.

[RKH01] Robles-Kelly, A., Hancock, E.R. (2001). Graph matching using adjacency matrix Markov chains. *Proceedings of the British Machine Vision Conference 2001.*

[RKH04] Robles-Kelly, A., Hancock, E.R. (2004). String edit distance, random walks and graph matching. *International Journal of Pattern Recognition and Artificial Intelligence*, v. 18(3), 315-327.

[SLP00] Schilling, C.H., Letscher, D., Palsson, B.O. (2000). Theory for the systematic definition of metabolic pathways and their use in interpreting metabolic function from a pathway-oriented perspective. *Journal of Theoretical Biology*, v. 203(3), 229-248.

[SB92] Shapiro, L.S., Brady, J.M. (1992). Feature-based correspondance: an eigenvector approach. *Image and Vision Computing*, v. 10, 283-288.

[SLH91] Scott, G. L., Longuet-Higgins, C. (1991). An algorithm for associating the features of two images. *Proceedings of the Royal Society of London, B: Biological Sciences*, v. 244, 21-26.

[S91] Steeb, Willi-Hans. (1991). *Kronecker Product of Matrices and Applications.* Bibliographisches Institut and F.A. Brockhaus AG, Mannheim.

[T04] Toran, J. (2004). On the hardness of graph isomorphism. *SIAM Journal on Computing*, v. 33(5), 1093-1108.

[TM69] Travers, J., Milgram, S. (1969). An experimental study of the small world problem. *Sociometry*, v. 32(4), 424-443.

[U76] Ullman, J.R. (1976). An algorithm for subgraph isomorphism. *J. Assoc. Computing Machinery*, v. 23(1), 31-42.

[U88] Umeyama, S. (1988). An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 10(5), 695-703.

[V03] Vespignani, A. (2003). Evolution thinks modular. *Nature Genetics*, v. 35(2), 118-119.

[WSK01] Wallis, W.D., Shoubridge, P., Kraetz, M., Ray, D. (2001). Graph distances using graph union. *Pattern Recognition Letters*, v.22(6-7), 701-704.

[WMF04] Wang, Y., Makedon, F., Ford, J. (2004). A bipartite graph matching framework for finding correspondences between structural elements in two proteins. *Proceedings of the 26th Annual Int'l Conf. of the IEEE Engineering Medicine and Biology Society*, San Francisco, CA.

[W99] Watts, D.J. (1999). *Small Worlds.* Princeton University Press, Princeton, NJ.

[W96] Wilson, R. J. (1996). *Introduction to graph theory.* Addison Wesley Longman Limited, England.

[XK01] Xu, L., King, I. (2001). A PCA approach for fast retrieval of structural patterns in attributed graphs. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, v. 31(5), 812-817.