ROLESIM AND ROLEMATCH:
ROLE-BASED SIMILARITY AND GRAPH MATCHING

A dissertation submitted to
Kent State University in partial
fulfillment of the requirements for the
degree of Doctor of Philosophy

by

Victor Eugene Lee

December 2012

Dissertation written by

Victor Eugene Lee

B.S., University of California, Berkeley, 1986

M.S., Stanford University, 1987

Ph.D., Kent State University, 2012

Approved by

Dr. Ruoming Jin , Chair, Doctoral Dissertation Committee

Dr. Javed Khan , Members, Doctoral Dissertation Committee

Dr. Robert Walker

Dr. Brian Castellani

Dr. Emmanuel Dechenaux

Accepted by

Dr. Javed Khan , Chair, Department of Computer Science

Dr. Raymond Craig , Dean, College of Arts and Sciences

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acknowledgments

I would first like to thank my advisor Dr. Ruoming Jin for guiding me through this five-year journey towards my doctorate. His intelligence, creativity, and hard work have been inspiring. I would also like to thank the other members of my dissertation committee, Javed Khan, Robert Walker, Brian Castellani, and Emmanuel Dechenaux, for their time, their probing questions, and their helpful comments.

With efficiency and cheerfulness, Marcy Curtiss and Sue Peti steered me through the administrative processes of the graduate computer science program at Kent State.

Hui Hong, Longjie Li, and Timothy Fox provided invaluable assistance with implementing and testing the algorithms in this dissertation. Learning is often a collaborative process, so I owe much to my fellow students and friends in our data mining research group, particularly to Ning Ruan and Lin Liu, whose years at Kent State have overlapped the most with mine.

To everyone who taught me and encouraged me throughout my life – but especially my family, my teachers and professors, and even employers – thank you. Special thanks go to my cadre of friends on Facebook, including but not limited to those in academia. They provided an ongoing source of advice, support, and humor.

The greatest and most heartfelt thanks go to my wife Susan. With the patience that is only possible through love, she endured years of my keeping odd hours and my periods of frustration and despondency. Having already attained her doctorate, she played the

multiple roles of spouse, best friend, proofreader, and academic counselor. I could not have completed this dissertation without her.

This dissertation is dedicated to my parents George and Sylvia Lee.

CHAPTER 1

Introduction

No man is an island, entire of itself;
Each is a piece of the continent, a part of the main;
If a clod be washed away by the sea, Europe is the less,
As well as if a promontory were,
As well as if a manor of thine own or of thine friend's were;
Each man's death diminishes me, for I am involved in mankind;
And therefore, never send to know for whom the bell tolls; It tolls for thee.

from *Meditation XVII*, John Donne [1]

Each of us is defined by our network of connections and interactions with others. Just as in Donne's time, we affect and are affected by those around us. Each individual is a part of the social group, and the social group is not the same without us. If this social view of individual-*cum*-corporate identity was true in Donne's 16th and 17th century England, it is even more so today. With the rise of the internet, mobile communications, electronic transactions, and personal broadcasting, the scale of connectedness has grown immensely. Not only can an individual interact with thousands and millions of others, but details about those interactions are being stored in databases, for later retrieval and analysis.

Network analysis is big business. In 2010, there were estimated to be over 100 software packages for network analysis, with IBM alone predicting that its annual sales for network analysis software would reach $15 billion by 2015 [2]. Moreover, the same techniques used to study human networks can be applied to studying any connected structure, such as an ecological web [3], interacting proteins [4], an electric power grid [5], the internet

itself [6], publications [7], or even words and concepts [8]. The potential for knowledge discovery seems unlimited.

How is it possible to analyze and understand such diverse and, in some cases, such large networks? Two key concepts help us to simplify and understand networks: structural patterns and social role. Networks often exhibit recurring structural patterns, and similar structure often correlates to similar functional or behavioral role. In his best-selling book *The Tipping Point*, Malcolm Gladwell describes three such recurring roles in social networks: the maven, the salesperson, and the connector [9]. Each helps to spread individual state changes throughout the network, but contributes to this process in a distinct manner.

Network analysis helps business users to learn more about customer interests and behavior, to tailor marketing efforts, to discover untapped resources and opportunities, and to uncover fraud or criminal activity. If businesses can classify customers into role-groups, they are better able to customize products and services to fit those groups. The presence of recurring roles and structural patterns also enables transfer learning: what we know about one network can be used to help us understand or identify information in another network. For example, matching points between graphs is a major approach for facial recognition in images [10]. Key points on the images can be automatically identified and connected to form a graph. If the topology of a subgraph of the image matches with a reference graph, then the subgraph may correspond to a face.

1.1 Computational Problems

Computational challenges remain, however. While the theoretical concept of structural roles is well-established [11, 12], and several definitions of role equivalence have been offered [13–17], there is no agreed-upon real-valued measure of role similarity. In this work, we focus on two specific computational problems: (1) developing a well-principled

and scalable measure for node structural similarity, and (2) finding the optimal node-to-node alignment between two graphs. We stress that in our problem, we do not begin with a pre-determined set of target roles. The roles are completely unknown at the start. Thus, our problem is akin to a clustering problem, grouping together items that are similar.

Role Similarity

Before we can make use of roles for uncovered information and knowledge in networks, we need to have a clear idea about what is a role. With real-world networks, however, we have a Goldilocks problem. If we chose a very exacting definition for roles, they we will almost never find two nodes with the same role. If we chose one that is too loose, then the role definitions lose significance. The answer is to measure role similarity. Two nodes may not have identical roles, but they might have similar roles. Have a similarity measure suggests that we should select a fairly strict definition of equivalence, and let the similarity measure handle inexact matches. We argue for selecting isomorphism, or when dealing with a single graph automorphism, as the basis of role equivalence. However, the graph isomorphism problem has no known polynomial time algorithm [18]. Furthermore, we still need to define a similarity measure, preferably one that is a metric. Thus we have two challenges, both ontological and computational: (1) What is a meaningful and tractable definition of role equivalence? (2) What is a meaningful and tractable definition of role similarity?

Approximate Graph Matching

Roles are given their meaning by their structural position within a graph. If two graphs are identical, then it is also true that each node has the same role as its partner in the other graph. If we have two similar but not identical graphs, can we use our understanding of roles to discover the best match between the graphs? If two nodes have similar roles, then they are good candidates for inclusion in the overall matching or alignment between the graphs. However, we once again have the graph isomorphism problem. Another question

concerns knowing when to stop searching for a match. If the two graphs are not identical, perhaps there is no good match for some nodes.

1.2 Driving Applications

Role-based network analysis will improve the effectiveness of many important network analysis applications, from market-driven customer analysis to scientific discovery. We take a look at two such applications: social recommendations and biomolecular network alignment.

1.2.1 Social Recommendation Systems

Role similarity answers the fundamental question: Who else is similar to me? Social networking sites, like Facebook, RenRen, and Google+, currently work based on direct links. However, while this been a successful model, the next step is to leap beyond direct connections and find someone else that fits a similar pattern or role. This may actually be the preferred model for marketers and advertisers. While certainly the friend-to-friend personal recommendation model is effective, businesses know that general consumer attitudes and behavior fits into categories akin to social roles, not dependent on direct links. For example, the mother in a family traditionally buys certain types of products that other mothers also buy, but which here own children do not buy. Businesses like Amazon are already using recommendations like"people who bought X also bought Y". This is still based on direct links: $prodA \leftarrow person \rightarrow prodB$. Role similarity offers a new holistic and global view.

1.2.2 Biomolecular Network Alignment

Biologists today are using a network approach to increase knowledge and understanding of biological systems. Many of us of have seen illustrations of predator-prey food chains and ecological webs, but molecular biologists are using networks too, to map out

biomolecular systems: biochemical pathway networks, gene regulatory networks, gene co-expression networks, and protein interaction networks. Proteins are the basic biochemical tools, regulators, and sometimes materials that drive and control all the varied processes to support life. Because each protein performs only a specific function, proteins must work together in process chains or complexes to accomplish more sophisticated and difficult functions. A protein-protein interaction (PPI) network describes all identified proteins and interactions. As of mid-2012, approximately 24 000 interactions among 5 000 proteins for *S. cerevisiae* (baker's yeast) have been identified [19]. However, while scientists have isolated many proteins and noted their interactions, there are still many unanswered questions. Taken out of context, the role of a protein is not always known or clear. Network analysis and evolutionary theory can help to answer these questions. Assuming all life forms evolved from a common ancestor, when we compare the networks of two or more species, we should find some similarities: similar subgraphs composed of similar proteins. These subgraphs represent portions of the PPI networks that have been conserved not altered by evolution. Then, anything we know about a conserved region in one network is potentially also true in another network. Moreover, where and to what degree we find similarity and alignment between networks is provides evidence for placing the branches in the evolutionary tree.

1.3 Dissertation Overview and Contributions

This dissertation makes the following contributions. First, to establish a sound theoretical basis, we present an axiomatic definition of a role similarity measure. This proves a clear and uniform understanding for the characteristics needed by any role similarity. The key axiom is automorphism/isomorphism confirmation: if two nodes are automorphically equivalent, then an admissible similarity measure must positively confirm this fact. Second, we present RoleSim, a role similarity metric which satisfies these axioms and which can be

computed with a simple iterative algorithm. RoleSim is founded on the concept of maximal matching of neighbor similarity. We rigorously prove that RoleSim satisfies all the axiomatic properties and demonstrate its superior interpretative power on both synthetic and real datasets. Third, we establish a recursive connection between local structural similarity and global network similarity, resulting in RoleMatch, a novel algorithm for matching two graphs. We demonstrate RoleMatch's effectiveness at matching not only very similar but also divergent graphs. RoleMatch is quite flexible, able to support graphs with weighted or directed edges, and with or without external information about node similarity. RoleMatch is also more scalable than other algorithms.

The dissertation is organized as follows. Chapter 2 surveys existing work on defining role equivalence, role similarity, and more general forms of node-level structural similarity. Chapter 3 develops a formal definition of role similarity. By considering the properties, strengths, and weaknesses of the various measures discussed in Chapter 2, we propose six axioms which define the proper attributes of a role similarity measure and metric. With none of the previous measures fully satisfying all the axioms, Chapter 4 introduces RoleSim, an axiomatically admissible role similarity metric. Chapter 5 presents an enhancement to improve the scalability of RoleSim. In Chapter 6, we introduce the maximal subgraph matching problem and develop RoleMatch to solve this problem. The scalability ideas of Chapter 3 are applied to provide an efficient implementation for RoleMatch. We offer concluding remarks in Chapter 7.

CHAPTER 2

Background and Related Work

This chapter provides a survey of prior work relevant to our core problem of role similarity. The first section describes several formal definitions of role and role equivalence. The next section reviews existing work on role similarity. The remaining sections review numerous measures for the general problem of local structural similarity, considering first centrality-based measures and then link-based measures.

We take this opportunity to establish some symbolic notation to used here and in the remainder of this dissertation. We use the terms *graph* and *network* interchangeably; the same is true for *vertex* and *node*. In most instances, we speak of networks and nodes, as this is the more common usage in the principal application domains of interest, but we revert to graph and vertex at times when speaking in a graph-theoretical sense.

We define a graph or network $G = (V, E)$ as a set of nodes V and a set of connecting edges $E \subseteq V \times V$. The neighbors of a node v are those nodes which are joined directly to v with an edge. The set of neighbors $N(v) = \{u|(u, v) \in E\}$. The degree of v is $d_v = |N(v)|$. When discussing computational complexity, the number of vertices in a graph is $n = |V|$, and the number of edges is $m = |E|$. By default, we assume that edges are undirected, but all of the concepts and formulas in this work can be extended to directed graphs by computing scores using in-neighbors and out-neighbors separately and then combining the scores.

2.1 Role Equivalence

Computing role similarity encompasses two more fundamental problems: what is a role, and how should we measure closeness to role? We use the following definition of

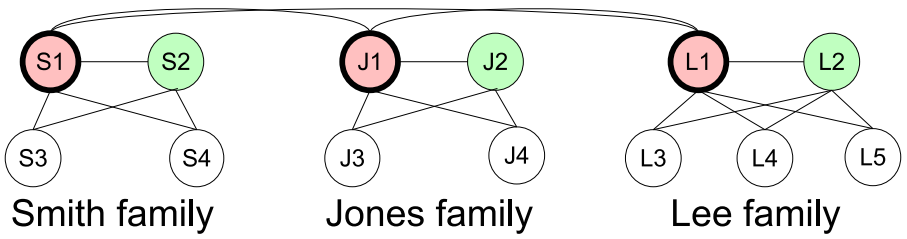


Figure 1: Example Graph for Role Equivalence

role:

Definition 1 Role and Role Equivalence. *A* role *is the set of relationships between an individual and others. In graph theory terms, the role of* v *is the set of all edges incident to* v*. For an undirected graph:* $role(v) = \{(u, v) \in E\}$*. Two individuals fulfill* equivalent roles *if they have equivalent relationships.*

For example, consider Figure 1, which depicts three siblings $\{S1, J1, L1\}$, who are each a parent in a family. Each family has two parents and either two or three children. There are three types of relationships shown:

1. Spouse {S1-S2, J1-J2, L1-L2}
2. Parent-Child {S1-S3, S1-S4, S2-S3, S2-S4, J1-J3, etc.}
3. Sibling {S1-J1, S1-L1, J1-L1}

For simplicity, we do not show the sibling relationships in the younger generation.

Intuitively, S1 and J1 appear to be role-equivalent: each is a spouse, a parent of two children, and the sibling of two others. Note we have not labeled or colored the edges, only the nodes. For example, a parent-child relationship is defined by the two participating nodes, not by a pre-labeling of the edge. However, we do not know that the two ends represent a parent and a child, until we identify the roles. In general, even the nodes will

Equivalence	**Neighbor Rule**	**Non-singleton Classes**	**Unique Partitioning?**
Structural	same nodes ($N(u) = N(v)$)	{S3,S4}, {J3,J4}, {L3,L4,L5}	Yes
Automorphic	For automorphism σ, $\forall\ x \in N(u),\ \exists\ y \in N(v)$ s.t. $y = \sigma(x)$	{S1,J1}, {S2,J2}, {S3,S4,J3,J4}, {L3,L4,L5}	Yes
Equitable Partition	same number per class	{S1,J1}, {S2,J2}, {S3,S4,J3,J4}, {L3,L4,L5}	No
Regular	same classes	{S1,J1,L1}, {S2,J2,L2}, {S3,S4,J3,J4,L3,L4,L5}	No

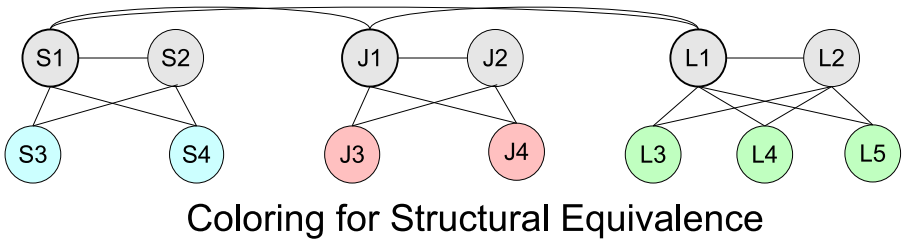
Table 1: Equivalence Classes for Figure 1

not be labeled or colored in advance. We will begin only with a graph topology; the role equivalence discovery problem is to identify the colorings.

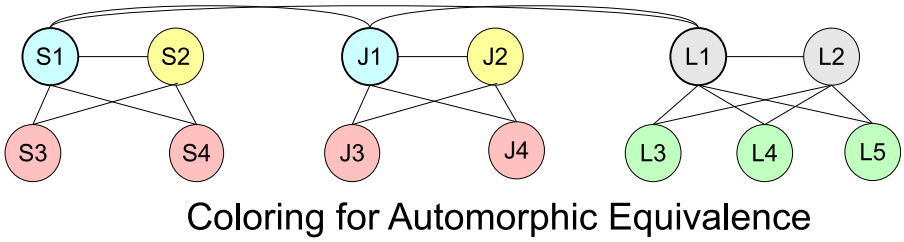
In social network analysis, the traditional approach for discovering role groups is to define a equivalence relation and to partition the actors into equivalence classes. Actors who fulfill the same role are equivalent. Over the years, four definitions have stood out. These four, in decreasing order of strictness, are structural equivalence, automorphic equivalence, equitable partition, and regular equivalence. Table 1 shows how these different definitions generate different roles from the same network.

• **Structural Equivalence:** Two actors are *structurally equivalent* if they interact with the *same* set of others [13]. Mathematically, u and v are structurally equivalent if and only if $N(u) = N(v)$. For example, consider the extended family shown in Figure 1. $S1$, $J1$, and $L1$ are siblings, $S2$, $J2$, and $L2$ are spouses, and the remaining nodes are their children. Each family's children, $\{S3, S4\}$, $\{J3, J4\}$, and $\{L3, L4, L5\}$, form a nontrivial equivalence class. However, none of the parents can be grouped together via structural equivalence. Figure 2(a) illustrates this partitioning. Nodes with the same color are in the same class, except gray nodes represent singleton classes. Each gray node is its own class. This model is too strict to be useful for simplifying a large network and to discover

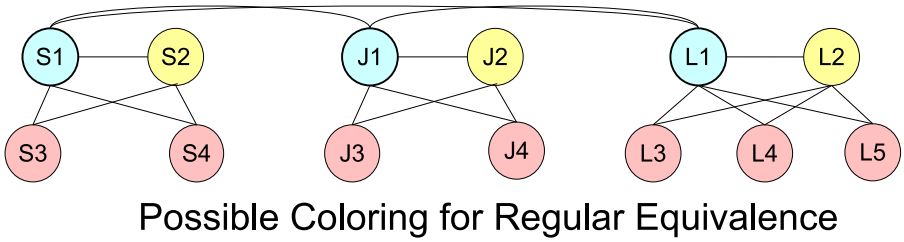
meaningful roles.



(a) Structural equivalence



(b) Automorphic equivalence



(c) Regular equivalence

Figure 2: Comparing Equivalence Schemes (Gray Nodes Are Not Equivalent)

• **Automorphic Equivalence:** Two actors (nodes) u and v are *automorphically equivalent* if there is an automorphism σ of G where $v = \sigma(u)$ [20]. An automorphism σ of a graph G is a permutation of vertex set V such that for any two nodes u and v, $(u, v) \in E$ iff $(\sigma(u), \sigma(v)) \in E$. In social terms, u and v can swap names, along with possibly some other name swaps, while preserving all the actor-actor relationships. Let $\Gamma(G)$ be the group of all automorphisms of graph G. For any two nodes u and v in G, $u \equiv v$ if

$u = \sigma(v)$ for some $\sigma \in \Gamma(G)$. Note that $\equiv$ is an equivalence relation on V; if $u \equiv v$ we say that u is automorphically equivalent to v. The equivalence classes generated under $\Gamma(G)$ (or $\equiv$) are called orbits. The equivalence class for vertex $v \in V$ is called the orbit of v, and denoted as $\Delta(v) = \{\sigma(v) \in V, \sigma \in \Gamma(G)\} = \{u | u \equiv v\}$. Each orbit corresponds to a role in the automorphic equivalence. Understanding the importance of automorphic equivalence and applying it to role modeling was a major breakthrough in classical social network research. In our example Figure 1, from the topology alone, we cannot distinguish between the Smith family and the Jones family. The Lee family is distinct, because it has three children instead of two. Therefore, the equivalence classes are $\{S1, J1\}, \{S2, J2\}, \{S3, S4, J3, J4\}, \{L1\}, \{L2\}$, and $\{L3, L4, L5\}$ (Figure 2(b)). Interestingly, automorphically equivalent classes must have equivalent indirect relations as well, such as equivalent in-laws and cousins. However, automorphic equivalence is hard to compute and still very strict.

• **Exact Coloration (Equitable Partition):** An *exact coloration* of graph G assigns a color to each node, such that any two nodes share the same color iff they have the same number of neighbors of each color [17]. Nodes of the same color form an equivalence class. An exact coloration is also referred to as equitable partition [21] and graph divisor [22] and is often applied in the vertex classification/refinement for canonical labeling in a graph isomorphism test [23, 24]. A graph may have several exact colorations; in general we seek the fewest colors. In our example, structural equivalence and automorphic equivalence offer two different exact colorations. Exact coloration relaxes automorphism by considering only immediate neighborhood equivalence, yet it still embodies a recursive aspect to role modeling.

• **Regular Equivalence (Bisimulation):** Two actors are *regularly equivalent* if they interact with the same variety of role classes, where class is recursively defined by regular equivalence [15]. Unlike automorphic equivalence and exact coloration, regular equivalence

does not care about the cardinality of neighbor relationships, only whether they are nonzero. For example, using regular equivalence, all three families could be equivalent, with only three equivalence classes: $sibling - parent\{S1, J1, L1\}$, $spouse - parent\{S2, J2, L2\}$, and $child$ (Figure 2(c)). Note that under regular equivalence, any two automorphically equivalent nodes may be merged into the same regular equivalence class. In computer science, the regular equivalence is often referred to as the bisimulation, which is widely used in automata and modal logic [25].

2.2 Existing Role Similarity Measures

We now move from strict equivalence to measuring similarity. There was been limited work on measuring role similarity. For structural equivalence, one can count how many neighbors they share, normalized by some factor. However, we noted in the previous section that structural similarity is too limiting for our interest.

Two algorithms for measuring the extent of regular equivalence are described in [16]. However, the authors acknowledge "the lack of a theoretical rationale for the measure of similarity produced." The core of the problem lies not in their algorithms, however, but in regular equivalence itself. Both regular equivalence and exact coloration are problematic because there may be more than one equivalence partitioning for a given graph. Indeed, for regular equivalence, every graph has two degenerate partitionings: (1) place all nodes in one class and (2) place each node in its own class (except structurally equivalence nodes may be in the same class). If one is measuring similarity, from which partition are you measuring the similarity?

To find the "best" regular partitioning, one can consider an information-theoretic or minimum description length (MDL) approach: group nodes into classes or blocks that approximately describe a true regular equivalence class membership. This is the blockmodeling approach [26, 27]. MDL blockmodeling tries to solve the following optimization

problem: Assign n nodes to b blocks such that the aggregate cost of describing the block structure $O(b \log b)$ plus the cost of describing the difference between the appropriate block structure and the true structure is minimized. Heuristically, the problem is easier if the number of blocks b is preset, but whether it is or not, the exact optimization problem is NP-hard. Furthermore, it does not truly address our problem: we want to know how similar are nodes at the individual level. Blockmodeling jumps ahead to a global partitioning problem and only provides a rough measure of distance.

2.3 Structural Similarity Measures

Due to the limited work in role similarity, we look at prior work for other types of structural similarity, namely, (1) centrality of a node with respect to the full graph and (2) link-based similarity. We will not consider density measures. Density has been well-studied in other works [28] and it not relevant to our final definition of role similarity.

2.3.1 Similarity of Node Centrality

This section discusses properties of individual nodes, in the context of a network. These properties can be interpreted as measuring some form of centrality, prestige, or authority. By themselves, these properties are not similarity measures. However, the property values, either scalars or vectors, of two different nodes can be compared to produce a similarity measure.

There are several simple measures of node centrality–the prominence of a node in the structural context of the graph. Each of these measures assigns a scalar value to a node. One can then compare how close the scores are between any two nodes. Node degree, closeness centrality, and betweenness centrality are three such measures. Degree counts the number of incident edges. In directed graphs, it can be divided into in-degree and out-degree.

Closeness centrality is the average distance between a node v and every other node in the graph. The classical measure of distance is shortest path distance, but other distances

have been proposed, such as random walk time [29] or the harmonic mean of all paths, not just shortest ones [30].

Betweenness centrality measures how frequently a node lies on the path between two other nodes [31]. Again, the classic notion uses shortest paths. If σ_{st} is the number of shortest paths between s and t, and σ_{st} is the number of short paths that include v, then the unnormalized betweenness centrality is

$$C_b(v) = \sum_{s \neq t \neq v} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Once again, it is possible to broader this to a random walk perspective: what is the likelihood that a random walker will pass through v on its way from s to t (a question many a roadside business has contemplated)?

These definitions, however, are too limited to encompass the concept of role. Role is not merely centrality of degree, centrality, or betweenness. It is any or all of these and possibly more; it is whatever makes the structural position of a node unique.

One way to extend the descriptive power of centrality measures is to use an ensemble approach: describing a node in terms of multiple attributes, in vector form. We already noted that degree can be split into in-degree and out-degree. Likewise, one could describe in terms of several centrality measures. For example, $C(v) = (C_{id}(v), C_{od}(v), C_c(v), C_b(v))$, with components for in-degree, out-degree, centrality, and betweenness, respectively. Then, to measure the similarity between weighted vector such as these, a common answer is cosine similarity:

$$S_c(a,b) = \frac{C(a) \cdot C(b)}{\|C(a)\| \, \|C(b)\|} = \frac{\sum_i C_i(a) C_i(b)}{\|C(a)\| \, \|C(b)\|} \tag{1}$$

Two more sophisticated measures bear mentioned: eigenvector centrality and graphlet

counting. In eigenvector centrality, the idea is that the importance of a node is the scaled sum of its out-neighbors' importances. Let the graph be represented by its corresponding $n \times n$ adjacency matrix A. Each cell value is either 0 or 1. Cell $a_{ij} = 1$ iff there is an edge from i to j. Then,

$$\begin{aligned} C_e(v) &= \frac{1}{\lambda} \sum_{u \in N(v)} C_e(u) \\ &= \frac{1}{\lambda} \sum_{u \in G} a_{vu} C_e(u) \end{aligned} \tag{2}$$

The simulation equation for all nodes can be written as the eigenvector equation $AC_e = \lambda C_e$. The eigenvector C_e is equal to the set of centrality measures when λ is the maximum eigenvalue. The eigenvector equation does not have an exact solution for all graphs. In such cases, an approximate solution will yield approximate centrality values.

A variant of eigenvector centrality is far better known: PageRank [32], which is interpreted as measuring authority. Using our notation, PageRank is defined as

$$C_{pr}(v) = \beta \sum_{u \in G} \frac{a_{uv}}{N(u)} C_{pr}(u) + \frac{1-\beta}{n} \tag{3}$$

PageRank reverses the direction from out-neighbors (a_{vu}) to in-neighbors (a_{uv}). More importantly, it divides each a_{uv} with $N(u)$, making the adjacency matrix into a stochastic transition matrix. We can then interpret our centrality values as probabilities. PageRank also adds a so-called dumping or jumping term $\frac{1-\beta}{n}$ which represents a uniform baseline probability of transitioning to any of the nodes. This guarantees that the eigenvector equation has a solution. All of these details can be smoothly interpreted with the random surfer model. Suppose the directed network is a road system, and a traveler begins at any random node. Each timestep there is a probability β that the traveler traverses a randomly selected out-edge from her current position, and a probability $1 - \beta$ that she magically jumps to any

random node. Over a long time, the PageRank score $C_{pr}(v)$ is the fraction of her total time that she spends residing at node v.

We mention one final measure, graphlet distribution. Graphlets are small induced unweighted subgraphs [33]. We can think of them as the structural primitives or building blocks of a graph. In practice, the size is set to be between 2 and 5 nodes, which provides 30 isomorphically distinct graphlets. There are 73 ways or orbits in which a given node v can intersect with one of these graphlets. So, for each node v, we count v's participation in each of the 73 orbits, producing a signature vector $\mathcal{G}(v)$. Pržulj *et al.* [33] proposes measuring the distance between two graphlet signatures as follows:

$$S_g(u,v) = 1 - \frac{\sum_{i=0}^{72} D_i}{\sum_{i=0}^{72} w_i}$$

where (4)

$$D_i(u,v) = w_i \frac{|log(u_i+1) - log(v_i+1)|}{log(max(u_i,v_i)+2)} \quad (5)$$

The weight w_i accounts for dependencies between orbits, described further in [33].

2.3.2 Link Similarity

Another way that node structural similarity has been defined is in terms of link similarity. That is, how are two nodes connected to one another? One of the earliest measures of link similarity is *bibliographical coupling* [34]. This measures the similarity between two research publications by counting the number of works that are listed in both of their bibliographies. *Co-citation* [35] turns this around by counting the number of later works that cite both of the two original documents. As the size of a work's bibliography increases, the likelihood that it will contain a particular work increases. Therefore, a common normalization of these two measures is to divide the count by the number of distinct works cited.

We can form a *citation graph*, where each node is a document and a directed edge (a, b) means that document a cites document b. Let $I(a)$ and $O(a)$ be the in-neighbor set and out-neighbor set of a, respectively. Let I_a and O_b be the in-degree and out-degree of a. Then, the normalized bibliographic coupling index is

$$S_{bc}(a, b) = \frac{|O(a) \cap O(b)|}{|O(a) \cup O(b)|}, \tag{6}$$

and the normalized co-citation index is

$$S_{cc}(a, b) = \frac{|I(a) \cap I(b)|}{|I(a) \cup I(b)|}. \tag{7}$$

These are simply the Jaccard coefficients of the out-neighbor sets and in-neighbors sets, respectively.

These two are suitable for unweighted and directed graphs. If a graph is undirected, then the two measures are the same. Suppose we have a weighted graph, though. This could be an author-collaboration graph, where edge (a, b) counts how many times author a has worked with author b. Or, it could be a bipartite document-term graph, where edge (d_a, t_b) counts the number of times that document a uses term b. Assign to each node a feature vector. For a co-authorship graph, each author is a feature dimension; its feature vector is the set of edge weights to every other author. For a document-term bipartite graph, a document has a term vector, weighted according to term frequencies of the document. If we represent the graph as an adjacency matrix, then the feature vector of node i is the i_{th} row of the matrix.

Given this representation, the cosine between two objects is a convenient and meaningful measure. Identical documents have cosine of 1, and documents with no features in

common are orthogonal with cosine of 0.

$$S_{cit}(a,b) = \frac{A \cdot B}{\|A\| \, \|B\|}, \tag{8}$$

where A is the feature vector of node a. While this measure bears superficial resemblance to Eq. 1 for centrality similarity, the difference lies in the composition of the vectors. In citation similarity, each dimension refers to one node. In centrality similarity, each dimension is an aggregate measure for the relationship between the node in question and the remainder of the graph.

A small modification to the denominator of Eq. 8, attributed to Tanimoto [36], maintains the overall behavior of the similarity function while aligning it with the Jaccard coefficient when the feature vectors are binary-valued:

$$S_{tani}(a,b) = \frac{A \cdot B}{||A||^2 + ||B||^2 - A \cdot B}, \tag{9}$$

Schultz [37] adapted the well-known TF-IDF query-document similarity measure to produce a term-weighted document-document similarity measure. Here, $A(t)$ is the frequency of term t for object a, and $idf(t)$ is the inverse document frequency for term t. More generally, it is the significance or importance of term t appearing in a document.

$$S_{wcos}(a,b) = \frac{\sum_{t \in T} A(t)B(t)idf(t)}{||A|| \, ||B||} \tag{10}$$

2.3.3 Iterative Link Similarity: SimRank and Extensions

Jeh and Widom [38] realized that a more general way to attack the node similarity problem was to not only look for shared neighbors, that is, neighbors that are *identical*, but to look for neighbors that are *similar*. This produces the recursive statement, "Two objects are similar if they are related to similar objects." [38] Formally, their SimRank measure is

defined as follows:

$$sim_{sr}(a,b) = \frac{c}{|I(a)|\,|I(b)|} \sum_{x \in I(a)} \sum_{y \in I(b)} sim_{sr}(x,y) \tag{11}$$

if $a \neq b$. If $a = b$, then $sim_{sr}(a,b) = 1$. c is a constant $0 < c < 1$. Also, for SimRank and all its variants, if either a or b has no neighbors, then $sim(a,b) = 0$. SimRank can be computed iteratively by initializing the matrix of $sim(.)$ values, hereafter called the S matrix, to the identity matrix.

Obviously, we can add the effects of in-neighbors and out-neighbors to produce a more comprehensive measure of the neighbor similarity between two objects. Several authors have proposed this [39, 40].

SimRank can be described as a recursive extension of the co-citation index. An important difference between the non-iterative algorithms in Section 2.3.2 and SimRank is that the earlier algorithms can be computed locally with a minimum of computational effort. With SimRank, however, to compute the similarity of even a single pair of objects, one has to consider the entire graph. This increases the computational requirements by a factor of n^2k, where k is the number of iterations. Consequently, several authors [41–44] have worked to reduce both the computational and memory requirements for SimRank, for general and specific applications.

In addition to concerns about the computational efficiency of the original SimRank formula, there are some structural flaws which mar its elegance. First, SimRank scores sometimes decrease when we would intuitively expect them to increase. Suppose we have an object-pair that has all neighbors in common. Then $sim_{sr}(a,b) = c/d$, d is the degree of a or b. As d increases, this should mean stronger ties between a and b, but clearly sim_{sr} actually decreases.

SimRank++

Antonellis *et al.* [45] partially compensates for this unwanted decrease by inserting an *evidence* factor. The more neighbors in common, the higher the evidence of similarity. They define evidence as

$$ev(a,b) = \sum_{i=1}^{|N(a)\cap N(b)|} \frac{1}{2^i}, \tag{12}$$

where $N(a)$ is the undirected neighbor set of a. If a and b have only one neighbor in common, $ev = 1/2$. As the number of neighbors increases, $ev \to 1$. This yields the following similarity definition:

$$sim_{ev}(a,b) = ev(a,b) \cdot c \sum_{x=1}^{N(a)} \sum_{y=1}^{N(b)} sim_{ev}(x,y) \tag{13}$$

The very narrow range $[0.5, 1]$ of the evidence factor, however, leads to the problem that $sim_{ev}(.)$ values are no longer bounded to a maximum of 1 or even to a constant. Instead, the maximum depends on the maximum value of $||N(a)|| \cdot ||N(b)||$ for the graph. The authors make one more extension to support edge-weighted graphs. Their final measure is called SimRank++:

$$sim_{spp}(a,b) = ev(a,b) \cdot c \sum_{x=1}^{N(a)} \sum_{y=1}^{N(b)} w_{ab} w_{by} sim_{spp}(x,y) \tag{14}$$

PSimRank

Fogaras and Rácz [46] realize that the cause of improper weighted of neighbor-matching in SimRank is due to the paired-random walk model. Ignoring the decay constant c for the moment, SimRank values are equal to the probability that two simultaneous random walkers, starting at nodes a and b, will eventually encounter each other. Even in the best case

scenario, in which a and b have all the same neighbors in common, so that $N(a) = N(b)$, the probability that the two walkers will happen to choose the same neighbor is $1/d_a$, which decreases as the degree increases. To emend this situation, Fogaras and Rácz introduce coupled random walks. They partition the event space into three cases:

1. Probability $P_1 = P($$a$ and b step to the same node$) = \frac{|I(a) \cap I(b)|}{|I(a) \cup I(b)|}$
2. Probability $P_2 = P($$a$ steps to a node in $I(a) \backslash I(b)) = \frac{|I(a) \backslash I(b)|}{|I(a) \cup I(b)|}$
3. Probability $P_3 = P($$b$ steps to a node in $I(b) \backslash I(a)) = \frac{|I(b) \backslash I(a)|}{|I(a) \cup I(b)|}$

Note that in case 1, which we would consider the direct similarity of a and b, is described by the Jaccard Coefficient. As required, the sum of these probabilities equals 1. We can then compute a similarity measure which takes the general form

$$sim_{ps}(a, b) = \sum_{i=1}^{3} P_i \cdot sim(\text{neighbors in Case } i).$$

Noting that there are $\frac{1}{|I(a) \backslash I(b)| \ |I(b)|}$ neighbor-pairs in Case 2 and $\frac{1}{I(b) \backslash I(a)| \ |I(a)|}$ in Case 3, this produces the logical but somewhat unwieldy formula:

$$sim_{ps}(a, b) = c \, [\, P_1 \cdot 1 + \frac{P_2}{|I(a) \backslash I(b)| \ |I(b)|} \sum_{\substack{x \in I(a) \backslash I(b) \\ y \in I(b)}} sim_{ps}(x, y) + \frac{P_3}{|I(b) \backslash I(a)| \ |I(a)|} \sum_{\substack{x' \in I(b) \backslash I(a) \\ y' \in I(a)}} sim_{ps}(x', y') \,]. \quad (15)$$

MatchSim

The authors of MatchSim [47] take this emendment of random walking to its limit. They observe that when a human compares the features of two objects, a human does not select random features to see if they match. Rather, people look to see if there exists an

alignment of features that produces a perfect or near-perfect matching. Therefore, their similarity measure discards the idea of random walk and replaces it with "the average similarity of the maximal matching between their neighbors." [47]:

$$sim_{ms}(a,b) = \frac{\sum_{(x,y)\in m^{\star}_{ab}} sim_{ms}(x,y)}{max(|I(a)|,|I(b)|)}, \tag{16}$$

where $m^{\star}$ represents the maximal matching. MatchSim omits the usual decay factor c, but this seems to be an idealization rather than a necessary alteration. Note that the size of the maximal matching is $min(|I(a)|,|I(b)|)$. Without loss of generality, assume a has fewer neighbors than b. The upper bound for $sim_{ms}(a,b)$ occurs when every neighbor of a is also a neighbor of b. In this special case, $max(sim_{ms}(a,b)) = max(\frac{min(|I(a),I(b)|}{max(|I(a),I(b)|}) = \frac{|I(a)\cap I(b)|}{|I(a)\cup I(b)|}$, which is the Jaccard coefficient.

2.3.4 Alternatives to SimRank

PageSim

All of the previous works are modifications of the original SimRank measure and principles. We now consider two measures that are markedly different than SimRank. We first consider PageSim [48], which not only borrows the entire PageRank computation as a starting point, but also borrows the meaning of PageRank's iterative computation to devise a related computation. The canonical interpretation of PageRank is that for each step, each page sends out an equal fraction of its own importance to each of its neighbors. Its importance for the next step is the sum of the fractional importance it received from its in-neighbors. PageSim also uses this spreading or propagating mechanism; however, rather than there being a universal importance feature which can be summed, each node begins

with a distinct self-feature, which is orthogonal to every other node feature. The authors describe the propagation process as occurring over distinct paths, and they sum the contributions of each path to compute the total distribution. As long as we permit self-intersecting paths, this is equivalent to measuring the random walk destination distribution for each node after k steps. PageSim follows a multi-step procedure:

1. For each node a, define feature vector $FV(a)$. $FV_b(a)$ is the b^{th} element of $FV(a)$.
2. Initialize all vectors: $FV_a^0(a) = PageRank(a)$. $FV_b^0(a) = 0, b \neq a$.
3. For $t = 1$ to k iterations, $FV^t = c \cdot \sum_{a \in V} \frac{FV^{t-1}(a)}{|O(a)|}$
4. Measure the similarity between pairs of feature vectors. In their original paper [48], the similarity measure is defined thus:

$$sim_{pg1}(a,b) = \sum_{i=1}^{n} \frac{min(FV_i(a), FV_i(b))^2}{max(FV_i(a), FV_i(b))} \tag{17}$$

In an expanded work [39], they modify the formula to more closely resemble the Jaccard coefficient:

$$sim_{pg2}(a,b) = \frac{\sum_{i=1}^{n} min(FV_i(a), FV_i(b))}{\sum_{i=1}^{n} max(FV_i(a), FV_i(b))} \tag{18}$$

Leicht's Vertex Similarity

The last measure that we consider addresses the other major weakness of SimRank: it considers only equal-length paths of similarity. As stated earlier, a SimRank value equals the probability that a given pair of nodes will meet *if they take steps simultaneously with the other.* That is, it would not count a case where Walker a takes 3 steps to reach c, and Walker b takes 4 steps to reach c. To address this limitation, Leicht *et al.* [49] formulate

their measure from the following maxim: "Vertex a is similar to b if a has any neighbor c this is itself similar to b." On one hand, this statement explicitly supports asymmetrical pairs of paths. On the other hand, it assumes that being neighbors implies similarity. In Leichts model, it follows that neighbors are somewhat similar, which describes clustering rather than role classification.

The authors did not give a catchy or convenient name to their measure, so for convenience we will call it VertexSim (notated sim_v or S_v). The initial version of VertexSim, written in matrix form is

$$\mathbf{S_v} = \phi \mathbf{A} \mathbf{S_v} + \mathbf{I}, \tag{19}$$

where A is the adjacency matrix and ϕ is a parameter to be determined. Solving for $\mathbf{S_v}$ and performing a power series expansion, we get

$$\mathbf{S_v} = \mathbf{I} + \phi \mathbf{A} + \phi^{\mathbf{2}} \mathbf{A^2} + \cdots.$$

After normalizing for the expected number of paths from a to b and some simplifying approximations, they authors finally derive the following:

$$\mathbf{S_v} = \mathbf{D^{-1}} \left(\mathbf{I} - \frac{\mathbf{c}}{\lambda_\mathbf{1}} \mathbf{A} \right)^{\mathbf{-1}} \mathbf{D^{-1}}, \tag{20}$$

where λ_1 is the largest eigenvalue of A, and D is the degree matrix (d_{ii} = degree of node i; all other $d_{ij} = 0$). Here we have a closed form solution, which seems convenient, but we also need to invert two matrices. An iterative computation process being simpler, the

Measure	Formula
bibliographic coupling	$S_{bc}(a,b) = \frac{\lvert O(a) \cap O(b) \rvert}{\lvert O(a) \cup O(b) \rvert}$
co-citation	$S_{cc}(a,b) = \frac{\lvert I(a) \cap I(b) \rvert}{\lvert I(a) \cup I(b) \rvert}$
cosine	$S_{cos}(a,b) = \frac{A \cdot B}{\lvert\lvert A \rvert\rvert \, \lvert\lvert B \rvert\rvert}$
Tanimoto	$S_{tani}(a,b) = \frac{A \cdot B}{\lvert\lvert A \rvert\rvert^2 + \lvert\lvert B \rvert\rvert^2 - A \cdot B}$
weighted cosine	$S_{wcos}(a,b) = \frac{\sum_{t \in T} A(t) B(t) idf(t)}{\lvert\lvert A \rvert\rvert \, \lvert\lvert B \rvert\rvert}$
SimRank	$sim_{sr}(a,b) = \frac{c}{\lvert I(a) \rvert \lvert I(b) \rvert} \sum_{x \in I(a)} \sum_{y \in I(b)} sim_{sr}(x,y)$
SimRank++	$sim_{spp}(a,b) = c \left(\sum_{i=1}^{\lvert N(a) \cap N(b) \rvert} \frac{1}{2^i} \right) \sum_{x=1}^{N(a)} \sum_{y=1}^{N(b)} w_{ab} w_{by} sim_{spp}(x,y)$
PSimRank	$sim_{ps}(a,b) = c \left(\frac{\lvert I(a) \cap I(b) \rvert}{\lvert I(a) \cup I(b) \rvert} + \frac{\sum_{\substack{x \in I(a) \setminus I(b), \\ y \in I(b)}} sim_{ps}(x,y)}{\lvert I(a) \cup I(b) \rvert \; \lvert I(b) \rvert} + \frac{\sum_{\substack{x' \in I(b) \setminus I(a), \\ y' \in I(a)}} sim_{ps}(x',y')}{\lvert I(b) \cup I(a) \rvert \; \lvert I(a) \rvert} \right)$
MatchSim	$sim_{ms}(a,b) = \frac{\sum_{(x,y) \in m^{\star}_{ab}} sim_{ms}(x,y)}{max(\lvert I(a) \rvert, \lvert I(b) \rvert)}$
PageSim	$sim_{pg2}(a,b) = \frac{\sum_{i=1}^{n} min(FV_i(a), FV_i(b))}{\sum_{i=1}^{n} max(FV_i(a), FV_i(b))}$
VertexSim	$DS_v D = \frac{c}{\lambda_1} A(DS_v D) + I$

Table 2: Structural Similarity Measures

authors rewrite the equation this way:

$$\mathbf{DS_v D} = \frac{\mathbf{c}}{\lambda_\mathbf{1}} \mathbf{A}(\mathbf{DS_v D}) + \mathbf{I}, \tag{21}$$

which we see resembles Eq. 19. The authors claim $\mathbf{DS_v D}$ can be initialized to any values such as $\mathbf{0}$ and will converge after 100 iterations or fewer.

We summarize the foregoing structural similarity measures in Table 2. We conclude this section with another table, which evaluates each of the aforementioned role equivalence or structural similarity measures, in light of how well it means our goal of a role similarity measure.

CHAPTER 3

Defining Axiomatic Node Similarity

In social science, it is well-established that individual agents tend to play roles or assume positions within their interaction network. For instance, in a university, each individual can be classified into the position of faculty member, administrator, staff, or student. Indeed, role discovery is a major research subject in classical social science [50]. Interestingly, recent studies have found not only do roles appear in other types of networks, including food webs [51], world trade [52], and even software systems [53], but also roles can help to predict node functionality. For instance, in a protein interaction network, proteins with similar structural roles tend to serve similar metabolic functions. Thus, if we know the function of one protein, we can predict that other proteins having similar roles would also have similar functions [54]. In other cases, such as online social networks, there are no *a priori* role categories. The classifications must be learned based on the interaction patterns alone.

In this chapter we tackle two problems. First, what are the necessary formal properties for a role similarity measure or metric? Second, how can we derive and compute a role similarity measure satisfying these properties? To address the first problem, we justify several axiomatic properties that define an appropriate role similarity measure or metric: range, maximal similarity, automorphic equivalence, transitive similarity, and the triangle inequality. For the second problem, we present RoleSim, a role similarity metric which satisfies these axioms and which can be computed with a simple iterative algorithm. We rigorously prove that RoleSim satisfies all the axiomatic properties and demonstrate its superior interpretive power on both synthetic and real datasets.

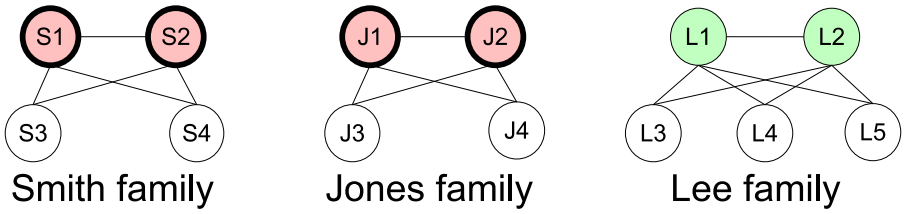


Figure 3: Role Equivalence of Unconnected Families

3.1 Automorphism-Based Role Similarity

A central question in studying the roles in a network system is how to define role similarity. In particular, how can we rank two nodes' role similarity in terms of their interaction patterns? We began Chapter 2 by defining our conceptual meaning for role equivalence. We then examined numerous measures for role equivalence and for general node similarity. Now, as we hone in on role similarity, we consider a corresponding statement about role similarity:

Two individuals fulfill similar roles *if they have* similar relationships *with others.*

This statement seems innocuous, but what it says and does not say are key to solving our problem. First, it is clear that we need to be more precise about the meaning of "similar relationships". In what ways can a set of relationships deviate from exact equivalence? They may vary in number and in kind. A parent having six children fulfills a similar but not identical role to a parent having two children. Roles may also vary by kind; however, in our case, we are focusing on the more challenging problem of unlabeled edges. The type of a relationship is determined not by the edge itself but by the roles-to-be-discovered of the two participants. Hence, we can clarify our statement to say the following:

Definition 2 Role Similarity. *Two individuals fulfill* similar roles *if they have a similar number of relationships with similar others.*

Note that we have now defined role similarity in terms of the similar of neighbors: a

Equivalence	Number of Neighbors	Connecting Path	Unique Partitioning
Structural	Equal	Required	Yes
Automorphic	Equal	Not required	Yes
Equitable Partition	Equal	Not required	No
Regular	Not Necessarily Equal	Not required	No

Table 3: Properties of Equivalence Classes

recursive definition. This is not unexpected, since we One thing our definition does not say is there should be any path joining similar individuals. This is key: role equivalence should not be based on being able to trace from one to the one. For example, the Smith family would still be topologically equivalent to the Jones family, even if they were unrelated. See Figure 3.

Table 3 summarizes the information from the Role Equivalence discussion (Section 2.1) in light of our expectations for similar number of relationships and similarity of neighbors.

From this we make two observations: First, for the boundary condition of perfect similarity, we would like to base our measure on automorphic equivalence. This is the only role definition that requires having the same number of relationships with the same type of neighbors, while having a unique configuration and not required connecting paths.

Thus, a key desire and computational challenge is to encapsulate graph automorphism into a role similarity metric. First, our measure should have a maximum value which corresponds to role equivalence. Due to the intractability of graph isomorphism discovery, existing role equivalence discovery algorithms [16, 26, 55] have relaxed the problem to equivalence confirmation: i.e., if any two nodes are automorphic, then the measure should yield the maximum value. The converse is not necessarily true; there may be some false positive assertions of equivalence. Regular equivalence can be interpreted as automorphic equivalence with false positives. Confirming automorphism is verifying a solution, which is often algorithmically less complex than discovering a solution.

However an equivalence rule can produce only binary similarity metrics: two nodes are

either equivalent (similarity $= 1$) or not (similarity $= 0$). In real-world networks, usually only a very small portion of the node-pairs would satisfy an equivalence criteria [56] and among those, many are simply trivially equivalent (such as singletons or children of the same parent). In addition, strict rule-based equivalence is not robust with respect to network noise, such as false-positive or false-negative interactions. It is desirable in many real world applications to rank node-pairs by their degree of similarity or provide a real-valued node similarity *metric*. Thus we have an open problem: *Can we derive a real-valued role similarity measure or ranking which complies with the automorphic equivalence requirement?*

3.2 Axiomatic Role Similarity

An equivalence relation tells us nothing about non-equivalent items. The real-world need is for a measure that not only recognizes automorphic equivalence, such as Smith child/spouse/parent to Jones child/spouse/parent (Figure 1), but also tells us that a Lee child is strongly similar to a Smith or Jones child, but not as similar to a Smith or Jones parent.

To deal with this shortcoming and to clarify the problem, we first identify a list of axiomatic properties that all role similarity measures should obey.

Definition 3 *(**Axiomatic Role Similarity Properties**) Given a graph $G = (V, E)$, any $sim(a, b)$ that measures the neighbor-based role similarity between vertices a and b in V should satisfy properties P1 to P5:*

- *P1) Range:* $0 \leq sim(a, b) \leq 1$*, for all a and b.*
- *P2) Symmetry:* $sim(a, b) = sim(b, a)$.
- *P3) Automorphism confirmation: If* $a \equiv b$*,* $sim(a, b) = 1$.
- *P4) Transitive similarity: If* $a \equiv b$ *and* $c \equiv d$*, then* $sim(a, c) = sim(a, d) = sim(b, c) = sim(b, d)$.

- *P5) Path independence: Whether* $sim(a, b) = 1$ *or not is independent of the path length from* a *to* b*.*
- *P6) Triangle inequality:* $d(a, c) \leq d(a, b) + d(b, c)$*, where distance* $d(a, c)$ *is defined as* $1 - sim(a, c)$*.*

Any node similarity measure satisfying the first five conditions (without triangle inequality) is called an **admissible role similarity measure**. *Any node similarity measure satisfying all six conditions is an* **admissible role similarity *metric***.

Property 1 describes the standard normalization where 1 means fully similar and 0 means completely dissimilar (i.e., the two neighborhoods have nothing in common). That this, we should always be able to recognize a purportedly equivalent node-pair by their similarity score of 1. Property 2 indicates that similarity, like distance, must be symmetric. Property 3 expresses our idea that fully similar means automorphically equivalent. Property 4 claims that the similarity between two nodes is equal to the similarity between any equivalent members of the first two node's respective equivalence classes. In other words, we can define the similarity between orbits: $sim(\Delta(u), \Delta(v)) = sim(u, v)$. This guarantees consistency of values at an orbit-level. Property 5 distinguishes role similarity from link-based or proximity-based similarity. Property 6 assumes the measure is metric-like, i.e., satisfying the triangle inequality. This is much stronger than transitivity, enforcing an *ordering* of values.

Note that Property 6 implies Property 4. (Let $b \equiv c$ so that $d(b, c) = 0$.) However, since most similarity measures do not necessarily satisfy the triangle inequality, we specify Property 4 separately. Further, Properties 3 and 5 are an essential criteria which distinguishes the role similarity measure from other existing measures. As we discussed earlier, the automorphic equivalence can be relaxed to exact coloration or regular equivalence. In that case, we may revise property P3 accordingly. This work will use the automorphic equivalence

axiom, though our framework can handle other forms of structural equivalence as well.

Theorem 1 *(**Generalized Transitive Similarity**) For any two pairs of nodes $a, b \in V$, $c, d \in V$, if $sim(a, b) = 1$ and $sim(c, d) = 1$, then, their cross similarities are all equal, i.e., $sim(a, c) = sim(a, d) = sim(b, c) = sim(b, d)$.*

Proof: From the triangle inequality, we have $d(a, c) \leq d(a, b) + d(b, c) \leq d(b, c)$ because $d(a, b) = 0$. Likewise $d(b, c) \leq d(b, a) + d(a, c) \leq d(a, c)$. Thus, $d(a, c) = d(b, c)$. Similarly, $d(a, d) = d(b, d)$, $d(c, a) = d(d, a)$, and $d(d, a) = d(d, b)$. Put together, we have $sim(a, c) = sim(a, d) = sim(b, c) = sim(b, d)$. □

Thus, if we partition the nodes into equivalence classes where the intraclass similarity equals 1, recording the similarity values between classes is sufficient to describe the similarity between any two individual nodes. Let $\Delta(x)$ and $\Delta(y)$ be the equivalence classes for node x and y, respectively. Then, we can define $sim(\Delta(x), \Delta(y)) = sim(x, y)$.

3.2.1 Binary-Valued Role Similarity Measures

Theorem 2 *(**Binary Admissibility**) Given any equivalence relation that also satisfies automorphism confirmation (P3) and path independence (P5), its binary indicator function is an admissible similarity* metric.

Proof: Binary values satisfy the Range requirement (P1). Any equivalence relation satisfies symmetry (P2) and transitivity (P4), by definition. For triangle inequality(P6), consider all possible cases:

Case 1: All in the same class: $0 \leq 0 + 0$

Case 2: All in different classes: $1 \leq 1 + 1$

Case 3: a and c in the same class: $0 \leq 1 + 1$

Case 4: b and one other in the same class: $1 \leq 0 + 1$

We have shown that a binary indicator function for an equivalence relation satisfies properties P1, P2, P4, and P6. Thus, if we are given that P3 and P5 are also met, then all properties are met. □

Note that automorphic equivalence, regular equivalence, and exact coloration all satisfy P3 and P5, so they are admissible metrics. Though these binary-valued similarity measures are admissible, they do not help us to understand the degree of similarity or dissimilarity. We would like a real-valued measure that ranks the degree of role similarity.

Before presenting our proposed real-valued role similarity metric for network roles, we first examine some similarity measures proposed in earlier works.

3.2.2 Similarity Measures That Are Not Axiomatically Admissible

Table 4 categorizes the previous chapter's similarity measures with respect to key axiomatic role similarity properties: Automorphism Confirmation (P3) and Path Independence (P5). We omit range (P1), symmetry (P2), and transitivity (P4) because they are not cause for rejecting any of these measures. Recall that the centrality measures (degree, closeness, etc.) by themselves are not similarity measures. We apply the Tanimoto coefficient to compare two centrality values to yield a similarity measure.

We also include an additional property, Degree Dependence, which is not axiom, but which may be considered desirable: If $d(a) \neq d(b)$, then $sim(a, b) < 1$. This property is complementary to Property 3. P3 requires that all automorphically equivalent node-pairs be identified, but it allows for false positives. Degree Dependence helps to reject some of these false positives by enforcing a simple property of all truly automorphic node-pairs. We can also view Degree Dependence as making the distinction between automorphic equivalence and regular equivalence.

In this table, only degree centrality, using the Tanimoto coefficient for comparison, satisfies all the selected properties. To see why the other centrality measures do not satisfy

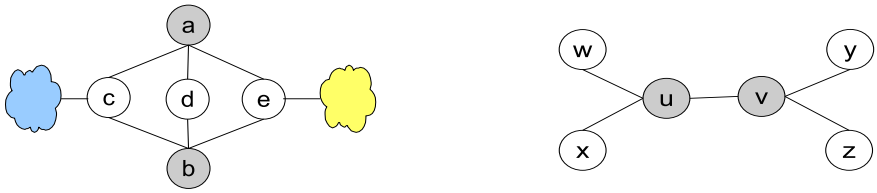
Similarity Measure	**Automorphism Confirmation**	**Transitivity**	**Path Independence**	**Degree Dependence**
Tanimoto(degree)	Yes	Yes	Yes	Yes
Tanimoto(closeness)	Yes	Yes	Yes	No
Tanimoto(betweenness)	Yes	Yes	Yes	No
Tanimoto(eigenvector)	Yes	Yes	Yes	No?
Tanimoto(PageRank)	Yes	Yes	Yes	No?
bibliographic coupling	Yes	Yes	No	Yes
co-citation	Yes	Yes	No	Yes
SimRank	No	–	No	–
SimRank++	No	–	No	–
PSimRank	No	–	No	–
MatchSim	Yes	Yes	No	Yes
PageSim	Yes	Yes	Yes	No?
VertexSim	No	–	No	–

Table 4: Properties of Similarity Measures

Degree Dependence, consider two star graphs, with different numbers of spokes. In each of the two stars, the hub nodes have the same closeness and betweenness, because every spoke node is exactly one link away. However, the two hubs are clearly not automorphically equivalent, due to the different degrees of the stars. The two PageRank-based similarity measures are thought to be inadmissible because it should be possible for two nodes to have the same PageRank scores without having the same local structure.

Bibliographic coupling and co-citation fail to meet the Path Independence property because they only count neighbors that are shared between the two nodes. Therefore, there must be a path of length 2 between the two nodes. The first iteration of SimRank is essentially the same as co-citation, counting in-neighbors in common, so SimRank and its variants are also not axiomatically admissible.

In addition, though SimRank seems to capture the intuition of recursive structural similarity, its random walk matching does not satisfy the basic graph automorphism condition. For example, in Figure 1, though $S1$ and $J1$ are automorphically equivalent, SimRank assigns them a value of 0.226. If the Smith (S1) and Jones (J1) families each had three



(a) Structural equivalence (b) Odd Distance

Figure 4: Problematic Configurations for SimRank

children, they would remain automorphically equivalent, but their SimRank score would decrease. We discuss this further in Section 3.2.2.

Taking a closer look at the computational behavior of SimRank will help us to see how to formulate a role similarity measure that is admissible. The SimRank similarity [38] between nodes u and v is the average similarity between u's neighbors and v's neighbors:

$$sim_{sr}(u,v) = \frac{(1-\beta)}{|N(u)||N(v)|} \sum_{x \in N(u)} \sum_{y \in N(v)} sim_{sr}(x,y), \text{for } u \neq v,$$

$$SR(v,v) = 1,$$

where β is a decay factor, $0 < \beta < 1$, so that the influence of neighbors decreases with distance. The original SimRank measure is for directed graphs. Here, we focus on its undirected version, though our comments also hold for the directed version. SimRank values can be computed iteratively, with successively iterations approaching a unique solution, much as PageRank [32] does.

Theorem 3 *SimRank is not an admissible role similarity measure.*

Proof: We give examples where property 3 (automorphic equivalence) does not hold. In Figure 4(a), a and b have the same neighbors. By even the strictest definition (structural equivalence), a and b have the same role. However, since SimRank's initial assumption is that there is no similarity among c, d, and e, when it computes the average similarity

of a and b's neighbors, it does not discover that a and b have equivalent neighborhoods. Assuming the best case where c, d, and e are in fact equivalent and using $\beta = 0.15$, $SR(a, b)$ converges to only 0.667. Even if the neighbors are not equivalent to one another, a to b should still be equivalent, but SimRank would give an even lower value. SimRank has an another problem (Figure 4(b)) when there is an odd distance between two nodes. Nodes u and v are automorphically equivalent, but because there are no nodes that are an equal distance from both u and v, $SimRank(u, v) = 0$!

We note that other variants of SimRank [40, 44–46, 57, 58] also do not meet the automorphic equivalence property for similar reasons. More discussion of these variants can be found in [59]. To our best knowledge, there is no available real-valued structural similarity measure satisfying the automorphic equivalence requirement.

CHAPTER 4

RoleSim: An Axiomatically Admissible Role Similarity Metric

To produce an admissible real-valued role similarity measure, we face two key challenges: First, it is computationally difficult to verify the automorphic equivalence property. Though not proven to be NP-complete, the graph automorphism problem has no known polynomial algorithm [18]. Second, all the existing real-valued role similarity measures have problems dealing with even simple conditions such as structural equivalence (Section 3.2.2). To meet these challenges, we take the following approach: Given an initial simplistic but admissible role similarity measurement for each pair of nodes, refine the measurement by expressing similarity in terms of neighboring values, while maintaining the automorphic and structural equivalence properties. Using this approach, we formally introduce RoleSim, the first admissible real-valued role similarity measure (metric) and its associated properties.

4.1 RoleSim Definition

Given a graph $G = (V, E)$, the RoleSim measure realizes the recursive node structural similarity principle "two nodes are similar if they relate to similar objects" as follows.

Definition 4 *(***RoleSim metric***) Given two vertices u and v, where $N(u)$ and $N(v)$ denote their respective neighborhoods and d_u and d_v denote their respective degrees, then*

$$RoleSim(u, v) = (1 - \beta) \max_{M(u,v)} \frac{\sum_{(x,y)\in M(u,v)} RoleSim(x, y)}{d_u + d_v - |M(u, v)|} + \beta \tag{22}$$

where $x \in N(u)$, $y \in N(v)$, and $M(u, v)$ is a **matching** *between $N(u)$ and $N(v)$, i.e., $M(u, v) = \{(x, y)|x \in N(u), y \in N(v),$ and no other $(x', y') \in M(u, v),$ s.t. $, x =$*

x' *or* $y = y'\}$. *The parameter* β *is a decay factor,* $0 < \beta < 1$.

The decay factor, similar to the one used in PageRank [32], both dampens the recursive effect and guarantees a minimal RoleSim score of β. We will sometimes abbreviate $RoleSim(u, v)$ as $R(u, v)$. $\mathbf{R}$ refers to the entire matrix of values. Figure 5 illustrates the matching process. Vertex u has three neighbors ($x_1, x_2, x3$), and v has four neighbors (y_1, y_2, y_3, y_4). The (x, y) grid is the subset of the RoleSim matrix of values corresponding to the pairings of neighbors of these two vertices. A matching selects one cell per row and column. If the number of rows differs from the number of columns, then the matching size is limited to $|M(u, v)| = min(d_u, d_v)$. A maximal matching is a matching where the total value of selected cells is maximum. In contrast, SimRank computes the average of every cell in the neighbor grid.

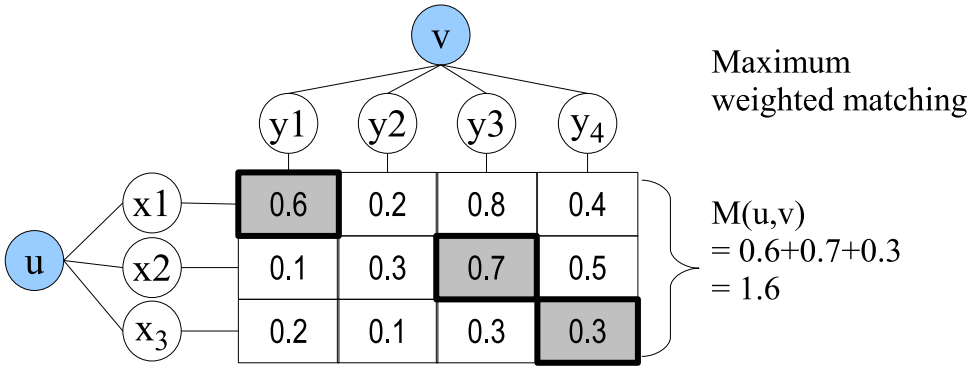


Figure 5: RoleSim(u,v) Based on Similarity of Their Neighbors

4.1.1 Relation to Jaccard Coefficient

RoleSim employs a generalization of the Jaccard coefficient, which measures the commonality between two sets A and B as $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$. Previous works [46] have used this index to compare node neighborhoods; several variants exist [60]. Our denominator is

similar to that of the Tanimoto coefficient [36], which measures similarity between multi-sets or between vectors. In our generalization, however, sets A and B are not vectors and need not share any common elements; instead, there is a weighted matching M between *similar* elements in A and B, i.e., $(a, b) \in M, a \in A, b \in B$. Let $r(a, b) \in [0, 1]$ record the similarity between a and b.

Definition 5 ***(*Generalized Jaccard Coefficient*)*** *The generalized Jaccard coefficient measures the similarity between two sets A and B under matching M, defined as*

$$J(A, B|M) = \frac{\sum_{(a,b)\in M} r(a, b)}{|A| + |B| - |M|} \tag{23}$$

The original Jaccard coefficient is a special case which uses the following matching M: Let $r(x, y) = 1$ if $x = y$; otherwise 0. Then define $M = \{r(x, x)|x \in A, x \in B\}$. Thus, the generalized Jaccard coefficient $J(A, B|M)$ reduces to $J(A, B)$. Comparing Eq. (22) and (23), we see that the heart of $RoleSim(u, v)$ is equivalent to the maximum of the generalized Jaccard coefficient between $N(u)$ and $N(v)$, among all matchings $M(u, v)$. Then,

$$RoleSim(u, v) = (1 - \beta) \max_{M(u,v)} J(N(u), N(v)|M(u, v)) + \beta \tag{24}$$

4.1.2 Relation to Weighted Matching

The definition and significance of $RoleSim(u, v)$ is closely related to *maximal weighted matching*. In our case, the matching is between the neighboring nodes of u and v.

Definition 6 Maximal Neighborhood Matching $\mathbb{M}(\mathbf{u}, \mathbf{v})$

Let $R(x, y)$ be a similarity score between any two nodes x and y (0 if no score exists). Given two nodes u and v, their neighborhood matching $M(u, v)$ is a weighted bipartite matching between neighbor sets $N(u)$ and $N(v)$ where the weights are the $R(x, y)$ scores.

The weight of the matching is $w(M) = \sum_{(x,y)\in M} R(x,y)$*. A maximal matching* $\mathbb{M}(u,v)$ *is an* M *with maximum weight.*

Using this, we can represent $RoleSim(u,v)$ in terms of maximal weighted matching $\mathbb{M}$. In Figure 5, the shaded cells represent the maximal matching: $0.7 + 0.6 + 0.3 = 1.6$.

Theorem 4 ***(Maximal Weighted Matching)*** *The RoleSim between nodes* u *and* v *corresponds linearly to the maximal weighted matching* $\mathbb{M}$ *for the bipartite graph* $(N(u) \cup N(v), N(u)\times N(v))$*, with each edge* $(x,y) \in N(u)\times N(v)$ *having the weight* $RoleSim(x,y)$*:*

$$RoleSim(u,v) = (1-\beta)\frac{w(\mathbb{M})}{\max(d_u, d_v)} + \beta \tag{25}$$

Proof: We need to show that Equations (22) and (25) are equivalent. Without loss of generality, let $d_u \geq d_v$. First, we show that *the cardinality of the maximal weighted matching* $|\mathbb{M}| = \min(d_u, d_v) = d_v$. It cannot be greater, because there are insufficient elements in d_v. It cannot be smaller, because if it were, there would exist an available edge between an uncovered node in d_u with one in d_v. Adding this edge would increase the matching (every edge has weight $\geq \beta$). If $|\mathbb{M}| = \min(d_u, d_v)$, it follows that $d_u + d_v - |M| = \max(d_u, d_v)$. Thus, the denominators in Equations (22) and (25) are constant and identical. It is then a trivial observation that the numerators are in fact the same. Therefore, the maximal value for the entire Equation (22) is the same as the value in (25). □

Theorem 4 not only shows the key equilibrium of role similarities between pairs of nodes in a graph G, but it also shows that RoleSim may be computed using existing maximal matching algorithms.

4.2 RoleSim Computation

RoleSim values can be computed iteratively and are guaranteed to converge, just as in PageRank and SimRank. First we outline the iterative procedure. In the next section, we prove that the calculated values comprise an admissible role similarity metric.

Step 1: Let the initial matrix of RoleSim scores be $\mathbf{R^0}$, estimated but admissible scores between any pair of nodes in G.

Step 2: Compute the k^{th} iteration $\mathbf{R^k}$ scores from the $(k-1)^{th}$ iteration's values, $\mathbf{R^{k-1}}$. Specifically, for any nodes u and v,

$$R^k(u,v) = (1-\beta) \max_{M(u,v)} \frac{\sum_{(x,y)\in M(u,v)} R^{k-1}(x,y)}{d_u + d_v - |M(u,v)|} + \beta \tag{26}$$

Based on Theorem 4, we compute Equation (26) by finding the maximal weighted matching in the weighted bipartite graph $(N(u) \cup N(v), N(u) \times N(v))$ with each edge $(x,y) \in N(u) \times N(v)$ having weight $R^{k-1}(x,y)$).

Step 3: Repeat Step 2 until $\mathbf{R}$ values converge for each pair of nodes in G.

Theorem 5 ***(Convergence)*** *For any admissible set of RoleSim scores $RoleSim^0$, the iterative computational procedure for RoleSim converges, i.e., for any (u,v) pair,*

$$\lim_{k\to\infty} RoleSim^k(u,v) = RoleSim(u,v) \tag{27}$$

This can be proven by showing that the maximum absolute difference between any $\mathbf{R^k}(u,v)$ and $\mathbf{R^{k+1}}(u,v)$ is monotonically decreasing. The complete proof is given in [59].

Unlike PageRank and SimRank which converge to values independent of the initialization, the convergent RoleSim score is sensitive to the initialization. Rather than being

a disadvantage, this sensitivity provides the necessary relaxation to compute automorphic role similarity in polynomial time, by utilizing the initialization as prior knowledge.

4.3 Admissibility of RoleSim

Here, we present one of the key contributions of this dissertation: the axiomatic admissibility of RoleSim. If the initial computation is admissible, and because the iterative computation of Equation (25) maintains admissibility (i.e., is an invariant transform of the axiomatic properties), then the final measure is admissible.

Theorem 6 ***(Invariant Transformation)*** *If the k^{th} iteration $RoleSim^k$ is an admissible role similarity metric, then so is $RoleSim^{k+1}$.*

For each axiomatic property P, we must show "If the k^{th} iteration $RoleSim^k$ satisfies Axiom P, then so does $RoleSim^{k+1}$." Properties 1 (Range) and 2 (Symmetry) are trivially invariant, so we will focus on the other four.

Automorphism Confirmation Invariance Proof: For nodes where $u \equiv v$, there is a permutation σ of vertex set V, such that $\sigma(u) = v$, and any edge $(u, x) \in E$ iff $(v, \sigma(x)) \in E$. This indicates that σ provides a one-to-one equivalence between nodes in $N(u)$ and $N(v)$. Also, u and v have the same number of neighbors, i.e., $d_u = d_v$. So, it is clear that the maximal weighted matching $\mathbb{M}$ in the bipartite graph $(N(u) \cup N(v), N(u) \times N(v))$ selects $d_u = d_v$ pairs of weight 1 each. Thus, $RoleSim^{k+1}(u, v) = (1 - \beta)\frac{w(\mathbb{M})}{\max(d_u, d_v)} + \beta = (1 - \beta)\frac{d_u \cdot 1}{d_u} + \beta = 1$. □

Transitive Similarity Invariance Proof: Assume transitivity holds for iteration k: for any $a \equiv b$, $c \equiv d$, $RoleSim^k(a, c) = RoleSim^k(b, d)$. Denote the maximal weighted matching between $N(a)$ and $N(c)$ as $\mathbb{M}$. Since there is a one-to-one equivalence correspondence σ between neighborhoods $N(a)$ and $N(b)$ and a one-to-one equivalence correspondence

σ' between $N(c)$ and $N(d)$, we can construct a matching $\mathbb{M}'$ between $N(b)$ and $N(d)$ as follows: $\mathbb{M}' = \{(\sigma(x), \sigma'(y))|(x, y) \in \mathbb{M}\}$. Since transitive similarity holds for $RoleSim^k$, we have $RoleSim^k(x, y) = RoleSim^k(\sigma(x), \sigma'(y))$. Thus, $w(\mathbb{M}') = w(\mathbb{M})$, and

$$(1-\beta)\frac{w(\mathbb{M})}{\max(d_a, d_c)} + \beta = (1-\beta)\frac{w(\mathbb{M}')}{\max(d_b, d_d)} + \beta$$
$$RoleSim^{k+1}(a, c) = RoleSim^{k+1}(b, d). \qquad \square$$

Path Independence Invariance Proof: If u and v are automorphic, then there is an automorphism which matches each neighbor of u with a neighbor of v. By Automorphism Confirmation Invariance, the similarity scores for each of these neighbor matchings will always be 1. Either the matching uses a path between u and v or it does not. If the current matching does not use a path (Case 1), then it will never need to use a path between them. Suppose that the current matching does use a path (Case 2). This can either be a path of length 1 (Case 2a: there is an edge (u, v)) or of length 2 (Case 2b: u and v have a common neighbor, called x). In Case 2a, we know that we can match $u - v$ because they are automorphic. Furthermore, we can hypothetically remove edge (u, v). This will reduce $N(u)$ and $N(v)$ by one member and reduce the matching size by 1, with no net effect on $R(u, v)$. So, Case 2a does not depend on the path. In Case 2b, it is not necessarily true that the automorphism matches $x \in N(u)$ to $x \in N(v)$. If it does, then we could hypothetically remove x, without affecting the score of $R(u, v)$, just as in Case 2a. If it does not use x, then it already does not depend on the path. Therefore, in all cases in which the current iteration makes use of a path connecting u and v, that path is not essential. If it is not essential now, in will not be essential in the next iteration. $\square$

Triangle Inequality Invariance Proof: For iteration k, for any nodes a, b, and c, $d^k(a, c) \leq d^k(a, b) + d^k(b, c)$, where $d^k(a, b) = 1 - RoleSim^k(a, b)$. We must prove that this inequality still holds for the next iteration: $d^{k+1}(a, c) \leq d^{k+1}(a, b) + d^{k+1}(b, c)$.

Observation: *if there is any matching M between $N(a)$ and $N(c)$ which satisfies $1 - ((1-\beta)\frac{w(M)}{d_c} + \beta) \leq d^{k+1}(a,b) + d^{k+1}(b,c)$, then $d^{k+1}(a,c) \leq d^{k+1}(a,b) + d^{k+1}(b,c)$.* This is because $\frac{w(M)}{d_c} \leq \frac{w(\mathbb{M})}{d_c}$, where $\mathbb{M}$ is the maximal weighted matching between $N(a)$ and $N(c)$, and thus, $1 - ((1-\beta)\frac{w(M)}{d_c} + \beta) \geq 1 - ((1-\beta)\frac{w(\mathbb{M})}{d_c} + \beta) = d^{k+1}(a,c)$.

We break down the proof into three cases:

Case 1. ($d_b \leq d_a \leq d_c$),

Case 2. ($d_a \leq d_b \leq d_c$), and

Case 3. ($d_a \leq d_c \leq d_b$).

Case 1: Since d_b is smallest, $|\mathbb{M}(a,b)| = |\mathbb{M}(b,c)| = d_b$. Define candidate matching M between $N(a)$ and $N(c)$ as $M = \{(x,z) | (x,y) \in \mathbb{M}(a,b) \wedge (y,z) \in \mathbb{M}(b,c)\}$. Then using our observation above:

$$
\begin{aligned}
& d^{k+1}(a,b) + d^{k+1}(b,c) - (1 - (1-\beta)\tfrac{w(M)}{d_c} - \beta) \\
&= (1-\beta)[-\frac{w(\mathbb{M}(a,b))}{d_a} - \frac{w(\mathbb{M}(b,c))}{d_c} + \frac{w(M)}{d_c}] + 1 - \beta \\
&= (1-\beta)[\frac{d_b - w(\mathbb{M}(a,b))}{d_a} - \frac{d_b}{d_a} + \frac{d_b - w(\mathbb{M}(b,c))}{d_c} \\
&\quad - \frac{d_b}{d_c} - \frac{d_b - w(M)}{d_c} + \frac{d_b}{d_c}] + 1 - \beta \\
&\geq (1-\beta)[1 - \frac{d_b}{d_a} + \frac{\sum_{(x,y)\in\mathbb{M}(a,b)}(1 - R^k(x,y))}{d_c} \\
&\quad + \frac{\sum_{(y,z)\in\mathbb{M}(b,c)}(1 - R^k(y,z))}{d_c} - \frac{\sum_{(x,z)\in M}(1 - R^k(x,z))}{d_c}] \\
&\geq (1-\beta)[\frac{\sum_{(x,y,z)}(d^k(x,y) + d^k(y,z) - d^k(x,z))}{d_c}] \geq 0
\end{aligned}
$$

where (x,y,z) means $(x,y) \in \mathbb{M}(a,b), (y,z) \in \mathbb{M}(b,c)$, and $(x,z) \in M$ $\square$

Cases 2 and 3 can be proven by a similar technique; the complete proof is in [59].

By combining the admissible initial configurations given in Sec 4.4 with Theorem 6 on invariance, we have shown that the iterative RoleSim computation generates a real-valued, admissible role similarity measure.

Theorem 7 *(***Admissibility***) If the initial* $RoleSim^0$ *is an admissible role similarity measure, then at each* k*-th iteration,* $RoleSim^k$ *is also admissible. When RoleSim computation converges, the final measure* $\lim_{k\to\infty} RoleSim^k$ *is admissible.*

4.4 Initialization

According to Theorem 7, an initial admissible RoleSim measurement $\mathbf{R^0}$ is needed to generate the desired real-valued role similarity ranking. What initial admissible measures or prior knowledge should we use? We consider three schemes:

1. **ALL-1** : $R^0(u, v) = 1$ for all u, v.
2. **Degree-Binary (DB)**: If two nodes have the same degree ($d_u = d_v$), then $R^0(u, v) = 1$; otherwise, 0.
3. **Degree-Ratio (DR)**: $R^0(u, v) = (1 - \beta)\frac{min(d_u, d_v)}{max(d_u, d_v)} + \beta$.

These schemes come from the following observation: *nodes that are automorphically equivalent have the same degree*. Basically, equal degree is a necessary but not sufficient condition for automorphism. This observation is key to RoleSim: degree affects both the size of a maximal matching set and the denominator of the Jaccard Coefficient.

Theorem 8 *(***Admissible Initialization***) ALL-1, Degree-Binary, and Degree-Ratio are all admissible role similarity measures. Moreover, Degree-Binary and ALL-1 are admissible role similarity* metrics.

Proof: It is easy to see that ALL-1 degenerately satisfies all the axioms of a role similarity metric. We focus on the two degree-based schemes. Clearly, they satisfy Range(P1)

and Symmetry(P2). If $N_u = N_v$, then $I(u, v) = 1$, so they both satisfy Automorphism Confirmation (P3). For transitive similarity (P4), we only need to show that $R^0(u, v)$ depends only on class membership (Theorem 1). For these schemes, class is defined by degree, and the measurement clearly depends only on degree. For Path Independence (P5), it is clear that the rules for choosing the initial values paid no attention to whether there exists a path between u and v. Finally, because Degree-Binary and ALL-1 are binary indicators of equivalence, Theorem 2 states that they are metrics. □

Note that SimRank's initialization ($SimRank^0(u, v) = 1$ iff $u = v$) is NOT admissible, because it sets the initial value of any potentially equivalent node-pairs to 0. SimRank iterations try to build up from zero. However, due to its problems with structural equivalence and odd-length paths that we noted, SimRank will never increase the value enough to discover equivalent pairs that were neglected at the start.

In addition, we make the following interesting observations on the different initialization schemes.

Lemma 1 *Let $\mathbf{R^1}(ALL - 1)$ be the matrix of RoleSim values at the first iteration after $\mathbf{R^0} = \mathbf{1}$ (All-1 initialization). Let $\mathbf{R^0}(DR)$ be the matrix of RoleSim initialized by the Degree-Ratio (DR) scheme. Then, $\mathbf{R^1}(ALL - 1) = \mathbf{R^0}(DR)$.*

This lemma can be easily derived by following the definition of RoleSim formula. Basically, the Degree-Ratio (DR) is exactly equal to the RoleSim state one iteration after ALL-1 initialization. Thus, ALL-1 and DR generate the same final results. The simple formula for DR is much faster than neighbor matching, so DR is essentially one iteration faster. On the other hand, we may consider the simple ALL-1 scheme to be sufficient, since it works as well as the more sophisticated DR. After the simple ALL-1 initialization, RoleSim's maximal matching process automatically discriminates between nodes of different degree and progressively learns the differences among neighbors as it iterates. Also, both ALL-1 and

DR initialization have the following convergence property:

Theorem 9 *(***Monotone Convergence***) If ALL-1 initialization is used, each RoleSim value is monotonically decreasing (or non-increasing):* $\mathbf{R^{k+1}}(u, v) \leq \mathbf{R^k}(u, v)$ *for all* k.

Proof: At any iteration, the RoleSim value for any (u, v) is the maximal matching of its neighbors. The value can increase only if some neighbor matchings increase. If no value increased in the previous iteration, then no value can increase in the current iteration. In the first iteration after ALL-1, clearly no value increases. Therefore, no value ever increases. □

Indeed, this monotone convergence property can be generalized into the following format: *if* $\mathbf{R^1} \leq \mathbf{R^0}$ *(that is, for every* (u, v) *pair,* $\mathbf{R^1}(u, v) \leq \mathbf{R^0}(u, v)$*), then* $\mathbf{R^{k+1}} \leq \mathbf{R^k}$. Note that the Degree-Binary (DB) initialization scheme does not have this property. In our experiments, we make an empirical comparison of these initialization schemes

4.5 Computational Complexity

Given n nodes, we have $O(n^2)$ node-pair similarity values to update for each iteration. For each node-pair, we must perform a maximal weighted matching. For weighted bipartite graph $(N(u) \cup N(v), N(u) \times N(v))$, the fastest algorithm based on augmenting paths (Hungarian method [61]) can compute the maximal weighted matching in $O(x(x \log x + y))$, where $x = |N(u) \cup N(v)|$ and $y = |N(u)| \times |N(v)|$.

A fast greedy algorithm offers a $\frac{1}{2}$-approximation of the globally optimal matching in $O(y \log y)$ time [62]. Furthermore, if an equivalence matching exists (i.e., $w(\mathbb{M}) = \max(d_u, d_v)$), the greedy method will find it. This is important, because it means that a greedy RoleSim computation still generates an admissible measure. Using greedy neighbor matching, the time complexity of RoleSim is $O(kn^2d')$, for k iterations, where d' is the average of $y \log y$ over all vertex-pair bipartite graphs in G. The space complexity is $O(n^2)$. In Chapter 4, we will introduce an approach for reducing both the time and memory cost.

4.6 Experimental Evaluation

In this section we experimentally investigate the ranking ability and performance of the RoleSim algorithm for computing role similarity metric values. We analyze the effect of different initialization schemes, and compare RoleSim to several state-of-the-art node similarity algorithms Specifically, we focus on the following:

1. How do different initialization schemes perform in terms of their final RoleSim score and computational efficiency?
2. Do node-pairs with high RoleSim scores have similar network roles, and for any two nodes known to have similar network roles, do they have high RoleSim scores?

To serve as reference models for our validation study, we utilize a well-known role-related random graph model and external measures of real datasets which provide strong role indication for these evaluations.

We set $\beta = 0.1$ for both RoleSim and SimRank, and we define convergence to be when values change by less than 1% of their previous values. We ran several RoleSim tests with both exact matching and greedy matching. The results were nearly identical: $> 90\%$ of node-pairs have the same score with the two methods, and the worst-case difference was small. Therefore, we focus on greedy matching from here on. We implemented the algorithms in C++ and ran all large tests on a 2.0GHz Linux machine with dual-core Opteron CPU and 4.0GB RAM.

For our tests, we use three types of graphs:

• **BL**: probabilistic block-model [63], where each block approximately corresponds to a role [64]. Here, nodes are partitioned into blocks. Each node in block i has probability p_{ij} of linking to each node in block j. Thus, the underlying block-model may serve as the ground-truth for testing role similarity.

• **SF**: Large Scale-Free random graphs [1] offer another model of large social or complex networks.

• Real-world networks, with a measureable feature similar to social role, are used for validating RoleSim performance.

4.6.1 Comparing Initialization

In Section 4.4 we saw that Degree-Ratio generates the same results as ALL-1 by short-cutting the first iteration. This reduces computation time by roughly 10%. Now we ask: Does Degree-Binary (DB) initialization (binary indicator which equals 1 when degrees $d_u = d_v$) give similar results, quickly?

We ran RoleSim using both ALL-1 and DB on 12 graphs, some scale-free and some block-model, having 500 to 10 000 nodes, and average node degrees from 1 to 10. We then converted RoleSim scores to percentile ranking, where 100% means the highest value, and 50% is the median value. Test results are summarized in Table 5. The three columns for Degree-Binary present the best, worst, and average performance among the 12 graphs. As noted earlier, Degree-Ratio generates exactly the same scores as ALL-1, albeit with one fewer iteration.

The high correlation coefficient values mean the rankings are virtually identical, so the rankings are not very sensitive to the initialization method. Moreover, DB took 20% from 68% less time to converge. Overall, DB seems to be the preferred initialization scheme in terms of computational efficiency. Thus, we adopt it for the rest of the experiments.

4.6.2 General Role Detection

How well does RoleSim discover roles in complex graphs? Specifically, given a ground truth knowledge of roles, do nodes having similar roles have high scores? To answer this, we generated probabilistic block-model graphs, where blocks behave like ”noisy” roles,

[1] http://pywebgraph.sourceforge.net/

Relative to ALL-1 Initialization	Degree-Binary			Degree-Ratio
	Min.	Avg.	Max.	
Difference in percentile rank	0.14%	0.38%	11.17%	none
Pearson correlation coefficient	0.9994	0.9998	0.9999	1
Relative execution time	0.32	0.52	0.80	≈ 0.9
Relative # iterations	0.38	0.58	0.88	1 fewer

Table 5: Comparison of Initialization Methods

due to sampling variance. We generated graphs with $N = 1000$ nodes and either 3 or 5 blocks. We varied the average node degree $\frac{|E|}{|V|}$, with higher values for graphs with more blocks. The size of each block and the p_{ij} values were randomized; we generated 3 random instances for each graph class. We compared RoleSim to the state-of-the-art SimRank, SimRank++ [45], and P-SimRank [46] measures.

For each measure and trial, we percentile-ranked its set of node-pair similarity scores. This normalizes the scoring among the four measures. Next, for each graph, we computed the average rank of all pairs of nodes within the same block, then averaged the three trials for each graph class.

Our results (Figure 6) show that RoleSim outperforms all other algorithms across all the tested conditions. None of the algorithms score perfectly, due to the inherent edge distribution variance of the probabilistic model. P-SimRank is better than SimRank, perhaps because it uses Jaccard Coefficient weighting, a step towards our RoleSim approach. Accuracy takes time. SimRank and SimRank++ run at the same speed. P-SimRank is about twice as slow, taking 184s to complete the most dense graph. RoleSim took 948s to complete the same graph.

4.6.3 Real Dataset: Co-author Network

We applied RoleSim and the best alternative measure, P-SimRank, to a real-world network having an external role measure. Our first dataset [65] is a co-author network of 2000 database researchers. Two authors are linked if they co-authored a paper from 2003 to 2008.

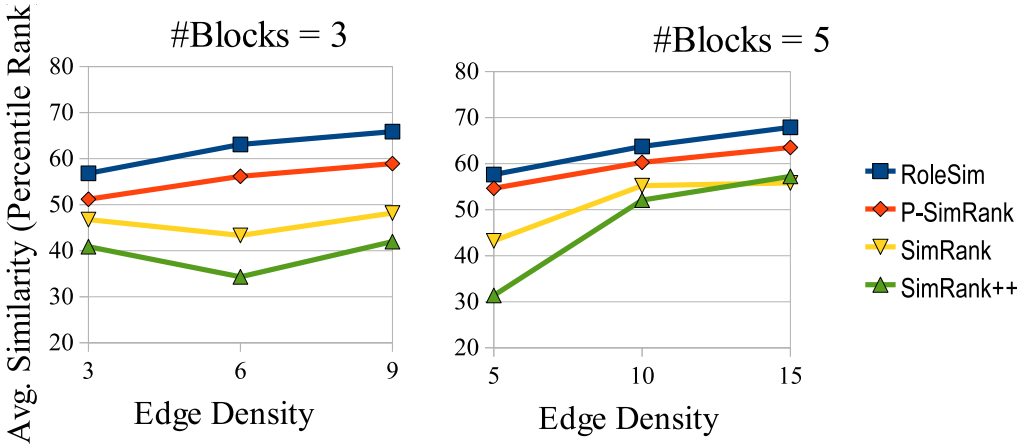
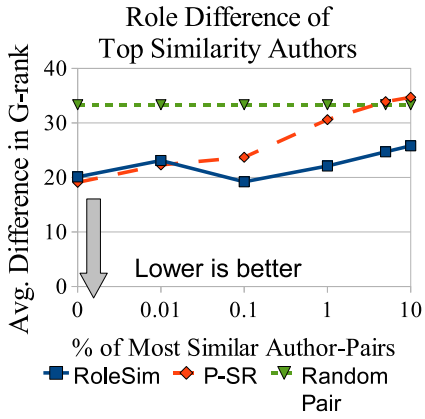


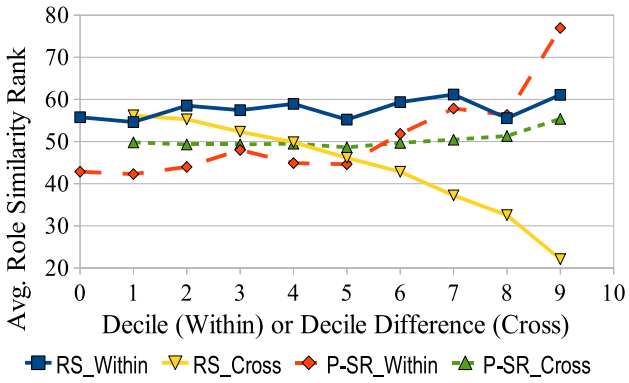
Figure 6: Average Similarity Ranking for Nodes in the Same Block

We pruned the network to the largest connected component (1543 nodes, 15483 edges). An author's role depends recursively on the number of connections to other authors, and the roles of those others. Hence, it measures collaboration. We use the G-index as a proxy measure for co-author role (H-index provides similar results and is omitted here). The G-index measures the influence of an academic researcher's publications, its value being the largest integer G such that the G most cited publications have at least G^2 citations. While G-index and co-author role are not precisely the same, G-index score is influenced strongly by the underlying role. High impact authors tend to be highly connected, especially with other high impact authors. If a paper is highly cited, this boosts the score of every co-author. Thus, we expect that if two authors have similar G-index scores, their node-pair is likely to have a high role similarity value. To normalize RoleSim, P-SimRank, and G-index values, we converted each raw value to a percentile rank.

Figure 7(a) addresses our second validation question (high rank$\rightarrow$ similar roles?). For the top ranked 0.01% of author-pairs, the average difference in G-index ranking is 20 points, for both RoleSim and P-SimRank, well below the random-pair difference of 33. A below-average difference confirms that the authors are relatively similar. However, as



(a) Top Coauthors



(b) Similarity of Authors Binned by K-index

Figure 7: Coauthor Role Similarity vs. G-Index Similarity

we expand the pool of author-pairs towards the top 10%, RoleSim continues to detect authors with similar authorship performance, while P-SimRank converges to random scoring.

To validate $role \rightarrow rank$ performance, we binned the authors into 10 roles based on G-index value (bottom 10%, next 10%, etc.). For every pair of authors within the same role decile, we looked up its role similarity rank and then computed an average per bin. We also computed averages for pairs of authors not in the same bin (dissimilar roles). Figure 7(b) shows our results. The average within-bin RoleSim value is consistently between 55% and

60%, better than the random-pair score of 50, and independent of whether the G-index is high or low; it performs equally well for all roles. P-SimRank within-bin scores (dashed line), however, are inconsistent. Performance of P-SimRank is worse than random for low G-scores, perhaps due to low density of links in the network. For the cross-bin data, the X-axis is the difference in decile bins for the two authors in a pair. The falling line of RoleSim indicates that role similarity correctly decreases as G-index scores become less similar. For P-SimRank, however, the cross-bin scores (dashed line) hover around 50, equivalent to random scoring.

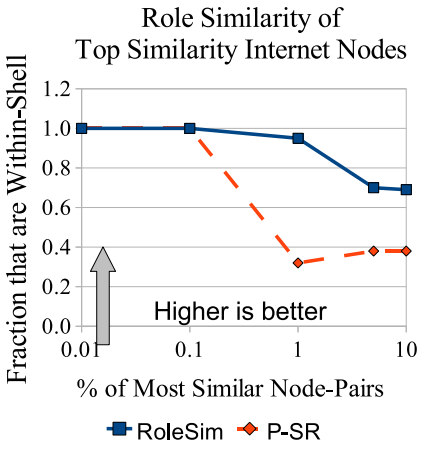
4.6.4 Real Dataset: Internet Network

Our second dataset is a snapshot of the Internet at the level of autonomous systems (22963 nodes and 48436 edges), as generated by Newman[2]. Several studies have confirmed that the Internet is hierarchically organized, with a densely connected core and stubs (singly-connected nodes) at the periphery [66, 67]. A node's position within the network (proximity to the core) and its relation to others affects its efficiency for routing and its robustness. Inspired by [67], we use K-shells to delineate roles.

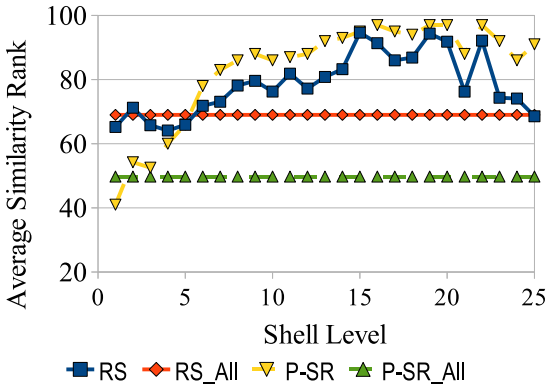
The K-core of a graph is the induced subgraph where every node connects to at least K other nodes in the subgraph. If $K' > K$, then the K'-core must be an induced subgraph of the K-core. The K-shell is defined as the 'ring' of nodes that are included in a graph's $(K-1)$-core but not its K-core. Thus we can decompose a graph into a set of nested rings, becoming denser as we move inward.

Using K-shells as our roles, we perform tests and analyses similar to those of the coauthor network. In Figure 8(a) we see that both measures do well for the top 0.1%, but P-SimRank's falters significantly when the range is expanded to the top 1%.

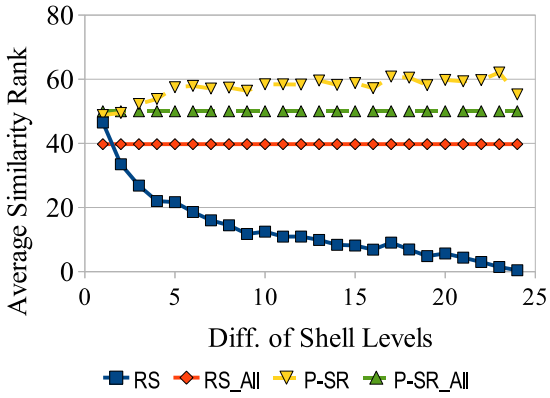
[2]Internet dataset, `http://www-personal.umich.edu/~mejn/netdata/`



(a) Similarity of Top Internet nodes



(b) Internet Graph Intra-Shell Similarity



(c) Internet Graph Cross-Shell Similarity

Figure 8: Internet Node Role Similarity vs. G-Index Similarity

Next, we treat K-shells the same way that we treated G-index decile bins in the previous test. See Figures 8(b) and 8(c). Unlike decile bins, the shells do not have equal sizes. K-shells 1, 2, and 3 together contain 92% of all nodes. To clarify how these three shells dominate, we also show horizontal lines representing the combined weighted average rank of all within-shell comparisons. RoleSim's within-shell values are consistently high, averaging 70%. Conversely, P-SimRank finds strong above-average similarity for the small high-K shells, but nearly random similarity for shells 1 to 3, pulling its overall performance

down to 50%.

In cross-shell analysis, RoleSim is able to distinguish different shells very well: RoleSim approaches zero as shell difference approaches maximum. On the other hand, P-SimRank shows almost no correlation to shell difference. Many of its scores are above-average when they should be below-average (dissimilar). On the whole, it seems that P-SimRank is not detecting role, but something related to connectedness and density.

In all these experiments, we can see that RoleSim provides positive answer to the role similarity ranking: 1) node-pairs with similar roles have higher RoleSim ranking than node-pairs with dissimilar roles, and 2) high RoleSim ranking indicates that nodes have similar roles. P-SimRank scores, however, do not correlate with network role similarity.

CHAPTER 5

Scalable Computation of Node Similarity

Node similarity ranking in general is computationally expensive because we need to compute the similarity for $\binom{n}{2} = O(n^2)$ node-pairs. A graph with $100,000$ nodes needs about 40GB memory to simply maintain the similarity values, assuming 8 bytes per value. Indeed, this is a major problem for almost all node similarity ranking algorithms. However, in most applications, we are interested only in the *highest* similarity pairs, which typically compose only a very small fraction of all pairs. Thus, in order to improve the scalability of RoleSim, we address the following challenge: *Can we identify the high-similarity pairs without materializing (storing) all the pair similarities?*

5.1 Iceberg RoleSim Computation

Formally, we consider the following question:

Definition 7 ***(*Iceberg RoleSim*)*** *Given a threshold θ, the Iceberg RoleSim problem is to discover all (u, v) pairs for which $RoleSim(u, v) \geq \theta$ and then to approximate their RoleSim scores.*

To solve Iceberg RoleSim, we consider a two-step approach: 1) use pruning rules to rule out pairs whose similarity score must be less than θ; and 2) apply RoleSim iterative computation to the remaining candidate pairs. Since RoleSim computation must match all $N(u) \times N(v)$ neighbor-pairs of a candidate pair (u, v), we have to handle some neighbor-pairs which are not themselves candidate pairs and therefore are not being stored. To address this need, we employ upper and lower bounds to estimate static RoleSim values for the non-candidate pairs.

Upper and Lower Bound for RoleSim:

Lemma 2 *Given nodes u, v and without loss of generality, assuming $d_u \geq d_v$, $R(u,v)$ is in the range $[\beta,\ \ (1-\beta)\frac{d_v}{d_u}+\beta]$.*

Proof: The definition of RoleSim is

$$R(u,v) = (1-\beta)\frac{w(\mathbb{M})}{d_u} + \beta$$

A lower bound for a neighbor matching weight $w(\mathbb{M})$ is 0. An upper bound is if each neighbor-pair in the matching has weight 1. Then the total weight is just the size of the matching $|\mathbb{M}| = d_v$. Plugging in our minimum and maximum values for $w(\mathbb{M})$ yields the range for $R(u,v)$. □

Lemma 3 *Given nodes u, v and without loss of generality, $d_u \geq d_v$, if $d_v \leq \theta d_u$, then similarity $R(u,v) \leq (1-\beta)\theta + \beta$. Equivalently, if we define $\theta' = (\theta-\beta)/(1-\beta)$, then if $d_v \leq \theta' d_u$, similarity $R(u,v) \leq \theta$.*

Proof: Replace $\theta = d_v/d_u$ in the upper bound range from Lemma 2. □

Lemma 3 tells us that without knowing any information other than node degrees, we can guarantee that the similarity scores for certain node-pairs will be below a threshold. With a little more calculation, a tighter bound and additional filtering can be achieved. We introduce the following iceberg pruning rules to filter out low scoring node-pairs. Without loss of generality, let $d_u \geq d_v$.

1. If $d_v < \theta' d_u$, then $R(u,v) < \theta$
2. If maximal matching weight $w(\mathbb{M}) < \theta' d_u$, then $R(u,v) < \theta$
3. Assume neighbor lists $N(u)$ and $N(v)$ are sorted by degree, with d_1^u and d_1^v being the degrees of the first items. The maximum possible similarity of this pair is $m_{11} =$

$(1-\beta)\frac{min(d_1^u,d_1^v)}{max(d_1^u,d_1^v)}+\beta$. If the shorter list has the smaller degree ($d_1^v \leq d_1^u$), and if $m_{11}+d_v-1<\theta' d_u$, then $R(u,v)<\theta$.

Rule 1 is just Lemma 3. Rule 2 is based on the upper bound of the RoleSim value. Rule 3 requires more explanation: continuing from Rule 2, we begin to consider all the pairings of neighbors. Because $N(v)$ is the shorter list, every member must contribute to the final matching. Either m_{11} will be in the matching or not. If it is, then an upper bound for $\mathbb{M}$ is if every remaining pair has weight 1, yielding $m_{11}+(d_v-1)$. Additionally, because the lists are sorted, $d_1^v/d_1^u \geq d_1^v/d_x^u$, for $x>1$. So, if m_{11} is too small to satisfy Rule 2, then all pairings using d_1^v are too small. This rule allows us to short circuit the full neighbor matching.

Algorithm 1 IcebergRoleSim($G(V,E)$, θ, β, α)

$H \leftarrow$ empty hash table indexed by node-pair ID (u,v);
$d_v \leftarrow$ degree of v;
Sort vertices V by degree;
for all $v \in V$ **do**
  $D^v=\{d_1^v,d_2^v,\cdots,d_{d(v)}^v\} \leftarrow$ degrees of neighbors of v, sorted by increasing order;
end for
for all $u \in V$ **do**
  for all $v \in V$ such that $\theta' d_u \leq d_v \leq d_u$ (Rule 1) **do**
    $m_{11} \leftarrow (1-\beta)\frac{min(d_1^u,d_1^v)}{max(d_1^u,d_1^v)}+\beta$;
    if $d_1^v \leq d_1^u$ and $N_v-1+M_{11}<\theta' N_u$ **then**
      Skip to the next v; (Rule 3)
    end if
    Compute maximal matching weight $w(\mathbb{M})$;
    if $w(\mathbb{M}) \geq \theta' d_u$ (Rule 2) **then**
      Insert $H(u,v) \leftarrow (1-\beta)w(\mathbb{M})/d_u+\beta$;
    end if
  end for
end for
Perform iterative RoleSim on H:
For neighbor pairs $\notin H$, use $\tilde{R}(x,y)=\alpha(1-\beta)N_x/N_y+\beta$

We now outline our approach, which is formalized in Algorithm 1. Rather than storing every node-pair's RoleSim score in an array, we store only the "tip of the iceberg" in a hash table. To generate the initial iceberg hash map, we first sort nodes by degree (line 3) and sort each node's list of neighbors, by degree (lines 4 to 6). The first sort allows us to consider only those node-pairs that are sufficiently similar in degree (line 8, pruning rule 1). We compute the estimated similarity for the first pair of neighbors. Note that this estimation formula is the same as Degree-Ratio initialization. If this weight is below the limit defined in Rule 3, we terminate this pair's candidacy and move on (lines 9 to 12). Otherwise, compute the remainder of neighbor-pair initial similarities, and perform a maximal matching. If the matching weight exceeds the θ' minimum bound (Rule 2), then this node-pair and its similarity are inserted into the hash table (lines 13 to 16). After iterating though all qualified node-pairs, we have our full hash table. We now perform RoleSim iterations, but only on members of the table, which typically is orders of magnitude smaller than a complete similarity matrix.

When a non-candidate pair's value is needed (as a neighbor-pair of a candidate pair), we apply the following estimate based on its lower and upper bound (assuming $d_u \geq d_v$):

$$\tilde{R}(u,v) = \alpha(1-\beta)\frac{d_v}{d_u} + \beta, \text{ where } 0 \leq \alpha \leq 1.$$

If $\alpha = 0$, $\tilde{R}(u,v)$ equals the lower bound; if $\alpha = 1$, the estimate is the upper bound. In the experimental evaluation, we will empirically study the effect of α on the estimation accuracy.

5.2 Performance of Iceberg RoleSim

In this experiment, we study how Iceberg RoleSim performs in terms of reducing computational time and storage, and its accuracy at approximating the RoleSim score for high

Average Density ($\lvert E\rvert/\lvert V\rvert$)	Iceberg Size, as fraction of full matrix	
	$\theta = 0.8$	$\theta = 0.9$
1	2.77%	1.47%
2	2.47%	0.63%
5	3.53%	0.15%

Table 6: Iceberg Size Relative to RoleSim Matrix

similarity node-pairs. Here, we generated 12 scale-free graphs with up to $100K$ nodes and edge densities of 1, 2, and 5. We compared standard RoleSim to Iceberg RoleSim, with θ values of 0.8 and 0.9. The parameter α, which is the weighting for estimated non-stored values, is set to midpoint 0.5. For scale-free graphs, the relative size of the iceberg hash table compared to the full similarity matrix depends on θ and edge density, but it is almost independent of the number of nodes. Table 6 shows that the icebergs' hash tables are only 0.15% to 3.5% as large as the full similarity matrices. Higher density graphs tend to have more structural variation and thus fewer highly similar node pairs. In Figure 9, we see that Iceberg RoleSim is an order of magnitude faster. To check that the ranking has not changed significantly, we computed the Pearson correlation coefficient for each graph's Iceberg RoleSim's rankings vs. the rankings from the corresponding portion of the full similarity matrix. For $\theta = 0.8$, the average coefficient is 0.823, and for $\theta = 0.9$, it is 0.880. Both show very strong correlation, indicating Iceberg RoleSim's very good accuracy at ranking role-similarity pairs.

Next we fixed θ at 0.9 and varied α from 0 to 1.0 to see how sensitive is the accuracy of Iceberg RoleSim with respect to α. The results from six scale-free graphs are shown in Figure 10. The labels describe the number of nodes and edges of each graph. Most graphs prefer $\alpha = 0$, but some prefer a midrange value. Any value in the lower half seems acceptable.

The Iceberg method of preselecting only the node-pairs that have the potential to be

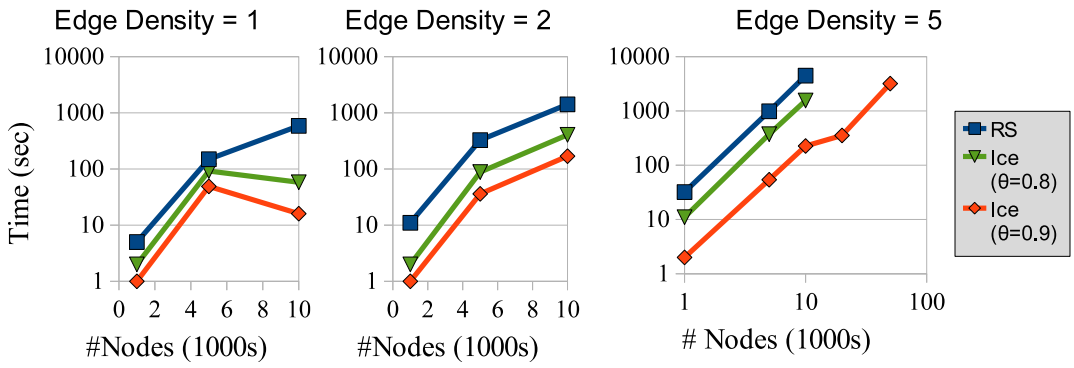


Figure 9: Execution Time: Standard vs. Iceberg

highly similarity is an effective way to improve the scalability of RoleSim while maintaining good similarity ranking. In the next chapter, we will discover that a similar approach can be used to improve the scalability of a more difficult problem: performing node-node and edge-edge matching of entire graphs.

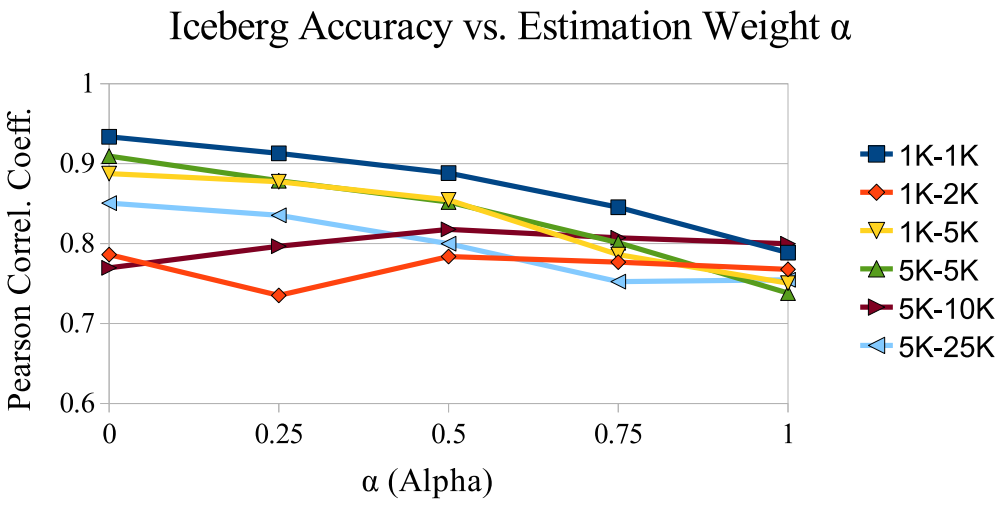


Figure 10: Iceberg Accuracy vs. α

CHAPTER 6

Role-based Alignment of Networks

Graph matching and comparison is an essential tool today for an ever-growing range of tasks, including image recognition, protein function discovery, and identity deanonymization. Finding the best node-to-node match between two differently-sized graphs is an ongoing challenge, since the subgraph isomorphism problem is NP-hard [68]. We approach the problem as maximal subgraph alignment: finding the largest set of nodes which have the greatest local similarity and which induce the greatest number of matched edges. We employ a recursive definition of structural similarity to develop an iterative method for computing a maximal alignment between two networks. At each iteration, we compute a set of local similarities, combine these into a tentative global alignment, and prune the set of local similarities. Our methodology is very flexible, supporting weighted edges, directed graphs, extended local neighbors, and inclusion of prior node similarity knowledge. We perform extensive tests on both synthetic and real-world data sets, demonstrating that RoleMatch meets or exceeds the performance of recent graph matching algorithms. In particular, RoleMatch is more scalable than any of the other algorithms examined.

6.1 Introduction

As network-structured data becomes commonplace, a natural question arises: are there hidden similarities between seemingly different networks? Is there a correspondence between nodes and edges of one graph to those of another, whether the nodes represent the same type of entities or not? In virtually all networks, the local structure around a node is related to the function or role performed by that node. Identity is tied to local network structure. Therefore, if we can detect similar substructures, this may indicate nodes that are

performing analogous roles.

Given two networks, how can we discover and match nodes which perform similar roles? That is, can we identify nodes which hold similar structural positions with their networks? Since role similarity implies a matching between local structures, can we extend node matching recursively to construct a global network-network matching?

Role-based matching holds the key to unlocking hidden knowledge in many network applications. An obvious application is to analyze a social network to find nodes that play similar social roles. For example, if a certain structural pattern represents a nexus of high influence in a product review network [69, 70], then we can search for similar substructures. Similarly, public health researchers can learn about the spread of disease by comparing social networks [71].

Network matching is also a powerful identity deanonymization tool. Given two related networks, one labeled and the other unlabeled (anonymized), we can use matching to predict the identities in the unlabeled network [72, 73].

A less obvious application is in molecular biology, where network alignment is used for transfer learning [74–79]. The functions of individual proteins can be hard to pinpoint. However, proteins tend to work together in modules to accomplish larger metabolic tasks, as modeled by a protein-protein interaction (PPI) network. By matching or aligning the PPI networks of different species, biologists can discover nodes and modules that may be related to one another through evolution. Any facts or features known about one node then become reasonable conjectures for its match mate.

We propose RoleMatch [80], a novel recursive role-based approach for finding the best matching subgraphs between two networks, based on both structural and node-level similarity. Using a recursive definition of role, it progressively learns the similarities between nodes and naturally expands from local (node) similarity to a more global (network) similarity. RoleMatch works with any kind of network, with or without directed edges, weights,

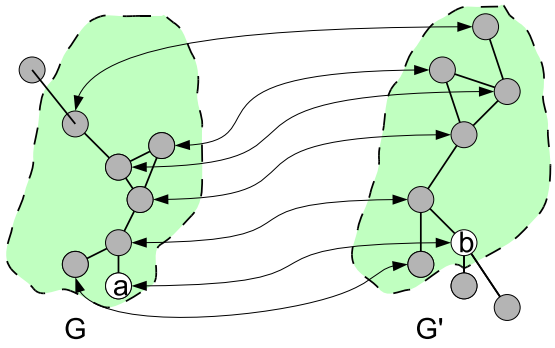


Figure 11: Network Alignment

or prior node similarity knowledge. It works fine without any hints or initial conditions, but it also can incorporate prior knowledge of node similarity, if provided.

These are two key intuitive ideas behind RoleMatch. The first is a recursive definition of local role similarity: two nodes match well if their neighbors can be matched well. The computation is implemented as an iterative algorithm. In each round, node-pairs compute a local role similarity score by computing a maximal matching of their neighbors' role similarities. The second idea is that the global network alignment is the natural extension of the local neighborhood matchings. The network alignment is a maximal matching of nodes and edges, so it is composed of an optimal selection of local matchings. Local matchings are constrained by the global alignment decisions which have already been made.

Figure 11 illustrates the basic idea of subgraph alignment. Note that within the context of the complete graphs, a and b (in white) have very different degrees ($d_a = 1$ and $d_b = 3$). However, within the subgraphs (shaded areas) they are isomorphic. Many existing methods measure structural similarity from the full graphs only, so they would have difficulty making this match. RoleMatch, on the other hand, incrementally prunes away portions that do not seem to match well and then reassesses the match quality of the remaining match candidates.

Figure 12 gives a closer idea of of how RoleMatch finds a maximal matching. The

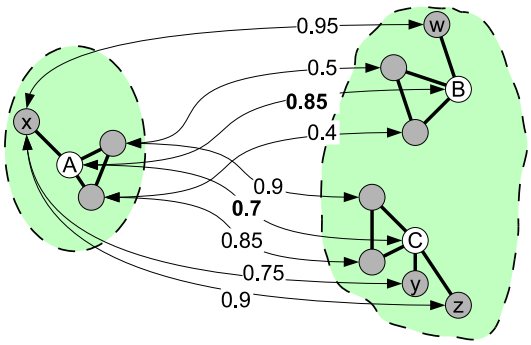


Figure 12: Alignment Based on Maximal Matching of Neighbors

values shown are a few of the node similarity scores from the current iteration. These may in fact be initial node label similarities. Similarity $s(A, B) = 0.85$ while $s(A, C) = 0.7$, so B seems to be better than C as a match for A. However, the best match between A's neighbors and B's neighbors have values $(.95, .5, .4)$, while the best match between A's neighbors and C's neighbors yields $(.9, .9, .85)$ which is clearly better. From X's perspective, even though W would be its best match, that would not be consistent with finding the best match for A. Since Y was not selected in the matching, it may be pruned. The next iteration's scores will be updated to incorporate the most recent maximal matching of neighbors, included the removal of Y.

RoleMatch is also scalable, employing efficient methods to prune the initial set of candidate matchings, greatly reducing the time and memory cost.

In the remainder of this chapter, we first review related work (Sec. 6.2). We then formalize our ideas on local matching, candidate pairs, and global matching (Sec. 6.3). Section 6.4 presents our complete algorithm. In Section 6.5, we verify the performance on several real and synthetic datasets.

6.2 Related Work

Role-based network alignment builds upon both node-level structural similarity and maximal subgraph matching. We look at past work in these two fields, and then we review recent work on node alignment in bioinformatics.

Structural Node Similarity: The key idea in structure-based node similarity is "two nodes are similar if their neighborhoods are similar." This statement has been quantified in numerous ways. SimRank [38] is recursively defined as the sum of the distance-decayed SimRank scores of all possible pairs of neighbors. This can be interpreted as the probability that two random walkers will eventually meet at a common node. In its original form, it is not suitable for comparing two separate graphs, since the walkers will never meet.

Several algorithms [33, 48, 73, 81] measure based on pure topological similarity, which allows them to compare two separate graphs. PageSim [48] labels each node with a unique feature, weighted by its PageRank score. It then disperses the features along outlinks for several iterations, records the net feature weights, and compares feature vectors to obtain a similarity score. ReFex [73] generates a structural feature vector for each node, based on the node degrees of its neighbors. Graphlets have also been used to construct node signatures [33]. Rather than using feature vectors, RoleSim [81] defines node similarity in terms of the maximal weighted matching between the sets of neighbors of the two nodes, where weights are recursively defined as RoleSim scores. This measure has been proven to positively confirm graph isomorphism: if the neighborhoods around two nodes are isomorphic, then RoleSim will discover the isomorphic matching.

Graph and Subgraph Matching: The graph isomorphism problem is to find a mapping, if it exists, between node sets V_0 and V_1 which preserves all the edges. That is, find a bijective function $\sigma : V_0 \rightarrow V_1$ for which $(u, v) \in E_0$ iff $(\sigma(u), \sigma(v)) \in E_1$. If the graph is labeled, then labels must match also. Unfortunately, there is no known polynomial solution to this problem [82]. While exact equivalence is rare for real-world graphs, the

isomorphism concept is important as the definition of the best possible match. We can relax the problem to look for the largest subgraphs of G_0 and G_1 that are isomorphically equivalent. This is the maximum common subgraph (MCS) problem, which is NP-hard. Several heuristics have been developed for the MCS Problem [83]. Another relaxation is from isomorphism to homomorphism. Fan [84] provides a guaranteed approximation algorithm for an injective homomorphic mapping from V_0 to V_1.

Real-world networks, however, rarely exhibit large-scale isomorphism. The task then is to find the best overall matching of nodes and edges that are similar to one another. One classic measure is graph edit distance. Here we seek the lowest cost sequence of node and edge deletions, label changes, and insertions which transform G_0 into G_1, given a set of unit costs for each fundamental editing operation [85]. The edit sequence implies a node alignment, for the nodes that are not added or deleted. The optimal solution has exponential complexity, so a number of suboptimal strategies have been developed [86]. Other approaches are need for large graphs. Zhu *et al.* [87] propose a heuristic of selecting anchor points that likely match, expanding, and then refining.

Several other methods reduce each graph to an alternate representation which summarizes its key features. These include spectral methods and graph kernel methods [88, 89]. However, because these summary representations are no longer node-based, they cannot satisfy our primary goal of aligning nodes. Riesen *et al.* [90] provide a good survey on exact and inexact graph matching.

Local and Global Network Alignment: The bioinformatics field has made many advances to the network alignment field in recent years. They classify alignments as either local or global. Local alignment algorithms such as PathBLAST [91], NetworkBLAST [92], and Graemlin [93] generally start by aligning one or more pairs of nodes based on label similarity only (ignoring graph connectivity initially), and then expand the alignment to neighboring nodes in greedy step-by-step fashion. However, these approaches assume that

label information (such as amino acid sequence and gene ontology) is available and of primary importance; these methods do not work for unlabeled graphs.

Global node alignment takes two basic forms. One approach is simply local alignment, when the expansion phase is required to continue until all nodes have been matching. Others taken a truly holistic global approach. Several works measure the matching quality as the number of edges that are conserved. Noting that each nonzero element in a graph's adjacency matrix represents an edge, PATH [94] and GA [95] seeks the permutation matrix which transforms graph G_0's adjacency matrix into the closest approximation of G_1's matrix. NATALIE [96] also uses matrix representation, framing the problem as integer linear programming. These matrix methods are not ideal for our maximal subgraph problem, because they only deal with complete matchings, which may not be appropriate when the two graphs not very similar.

The following methods are the most similar to RoleMatch, because they also employ a structural node similarity measure as the foundation for a maximal alignment. They differ from RoleMatch in their objective functions for measuring alignment quality, their node similarity measures, and their method for discovering a maximal alignment. IsoRank [97] takes a hybrid two-phase approach. First, an iterative formula generates a similarity score for all possible node-pairs. The formula embodies the idea that the similarity between a pair of nodes is based on the weighted similarity of their neighbors. The author's compare it to PageRank, but in fact it is closer to SimRank [38], with the addition of prior label similarity information. This prior information is essential to establish an initial connection between the otherwise separate graphs. Thus, though it generates node-level scores, the scores depend on the entire graph's topology. In the second phase, the seed-grow approach of local alignment is applied. IsoRank does not attempt to maximize any objective function. Instead, alignments are allowed to growing as long as new members satisfy a similarity constraint relative to the seed node-pair. GRAAL [98] assigns to each node a feature vector,

which records the number and type of adjacent graphlets. MI_GRAAL [99] also records node degrees, clustering coefficient, and eccentricity. They then use some version of seed-and-grow to construct an alignment.

6.3 Subgraph Alignment

Our basic goal is to find (1) the largest subgraphs of a pair of graphs that have (2) the highest structural similarity to one another. For the first issue, it is quite straightforward to measure the size of subgraphs, by counting nodes or edges. The second issue is more challenging: How does one quantify structural similarity? We begin by formalizing our problem.

In addition to and independent of its neighbor relationships, a node may have features or labels. Let $f(v)$ denote the feature vector for node v. For example, in a protein-protein interaction network, each protein may be labeled with gene ontology and protein structure information. F_{uv} is a feature similarity function which compares $f(u)$ to $f(v)$. Similarity scores are normalized to the range 0 to 1: $F_{uv} = 1$ when $f(u) = f(v)$.

Definition 8 Subgraph Matching $\mathbf{M(G_0, G_1)}$

Given two graphs $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$, a subgraph matching *M is a bipartite matching between a subset of V_0 and a subset of V_1. The symbols V_0^M and V_1^M indicate the subsets of V_0 and V_1, respectively, covered by M. Furthermore, the subgraphs induced by V_0^M and V_1^M are called $G_0^M = (V_0^M, E_0^M)$ and $G_1^M = (V_1^M, E_1^M)$.*

Definition 9 Maximal Subgraph Alignment Problem

Let G_0 and G_1 be two graphs. Given a function $S_M(H_0, H_1)$ that measures the similarity between two graphs, determine a subgraph matching M that maximizes the following

objective function:

$$A_M(G_0, G_1) = S_M(G_0^M, G_1^M) \left(\frac{|V_0^M| + |V_1^M|}{|V_0| + |V_1|} \right)^{1/\alpha} \tag{28}$$

where $\alpha \geq 1$. The latter term measures the size of the subgraphs, relative to the original graphs. The exponent $\frac{1}{\alpha}$ controls the importance of subgraph size compared to subgraph similarity. We balance the benefits of including strongly similar nodes (first term) against the drawback of not covering the full graphs (second term). Higher values of α will seek smaller but very strongly similar subgraphs. In general, we can relax M from a bipartite matching to be a multiway matching.

6.3.1 Graph Similarity Function

We now explore formulations for $S_M(G_0^M, G_1^M)$. Isomorphism represents perfect similarity, but how do we quantify the similarity when the graphs are not identical? One classic approach is, given a node-node mapping, to count the number of induced edges that match. If the mapping is an isomorphism, then all edges will match. While this approach is reasonable in many cases, we discovered a weakness during experimentation. An edge-driven method, such as [94] or [95], might match a high percentage of edges but a low percentage of nodes. This may occur if a graph contains several dense regions. By simply matching any dense cluster to any other one, we may tabulate a high number of edge correspondences. If the low density regions do not match well, it does not affect the score very much, hence a edge-driven method has low incentive in this type of graph to match the global topology correctly.

We reduce the problem of global similarity to a recursive local similarity problem. Our approach it to consider a graph as the composite of all the local neighborhoods around every node, just as a city can be described by all its neighborhoods. Each node may have its own

characteristics ($f(v)$ features), but how it connects to other nodes is equally important. This is clear, when one considers that one of the standard ways of encoding a graph is as a list of direct neighbors (an adjacency list) for each node. There is the danger, however, of being "shortsighted" by only considering immediate surroundings. Consequently, we apply our measure of neighborhood similarity recursively, so that the global structure influences the local similarity and vice versa.

There are several recursive measures of local structural similarity ([33, 38, 48, 73, 81]). RoleSim [81] is particularly well suited for our needs because it is simple yet powerful enough to affirm graph isomorphism. This simplicity allows it Next we make a simple modification, to introduce the node feature similarity F_{uv} as well. We then consider how to improve the scalability, while preserving our overall goal of obtaining a high quality graph matching. The core of local similarity is maximal neighborhood matching $\mathbb{M}$, from Definition 6 in Chapter 4.

This leads to a complete definition of local similarity:

Definition 10 Recursive Local Similarity $\mathbf{R(u, v)}$

Given a pair of nodes u and v, the local similarity score $R(u, v)$ is a weighted sum of their feature similarity F_{uv} and a maximal bipartite neighbor matching $\mathbb{M}$. Specifically,

$$R(u,v) = \beta F_{uv} + (1-\beta)\frac{\sum_{(x,y)\in\mathbb{M}(u,v)} R(x,y)}{max(d_u, d_v)} \tag{29}$$

where $0 \leq \beta \leq 1$. A perfect score is 1. If node features are not being considered, then F_{uv} should be set to 1 for all node pairs, and the equation degenerates to RoleSim.

Note that the definition of $R(u, v)$ does not specify that u and v belong to the same graph. So, we can iteratively compute *cross*-graph scores, as long as we have an initial set of scores. Another attractive feature of RoleSim is that in can be initialized by setting all $R^0(u, v) = 1$.

Measuring the similarity between graphs follows naturally from RoleSim's neighbor matching. This local similarity measure accounts for feature similarity, edge matching, and structural similarity. Thus, to measure the quality of a global or subgraph matching, we simply need to find a good matching of nodes and then sum all the $R(u, v)$ scores in the matching. In practice, we use the average instead of the sum, in order to normalize the global score, so that isomorphic graphs have a score of 1.

6.3.2 Qualified Pairings

Given a set of cross-graph $R(u, v)$ scores, finding the maximal graph alignment is a straightforward case of maximal bipartite matching. However, for large graphs, the computational complexity is daunting. We need to reduce the number of candidate pairs. Pruning away node-pairs that are poor matches has a two-fold benefit: It simplifies global graph alignment, but it also simplifies local node-to-node scoring. Consider Figure 13. Node u has 3 neighbors and v has 4 neighbors. To find the maximal neighbor matching, we have 12 possible pairs to consider. Now suppose that the white background cells, such as (x_1, y_2), are considered poor matches, possibly because of their low $R()$ scores. we can eliminate them from consideration. In this example, u and v only need to consider 7 (shaded) cases instead of 12. Note that this pruning of candidates for local alignment is simultaneously pruning candidates for global alignment. By eliminating poor pairings, we are driving towards the desired 1-1 matching.

We formalize this idea in terms of *qualified pairings*. We use the word *pairing* to emphasize that a pair of nodes $(u, v), u \in V_0, v \in V_1$ is a candidate for matching, but it may or may not be selected for use when seeking a maximal matching. To ensure consistency between global alignment and local similarity, we only permit globally qualified pairings to be considered when matching local neighbors.

Definition 11 Qualified Pairs, Nodes, and Neighbors: *Given graphs G_0 and G_1, the*

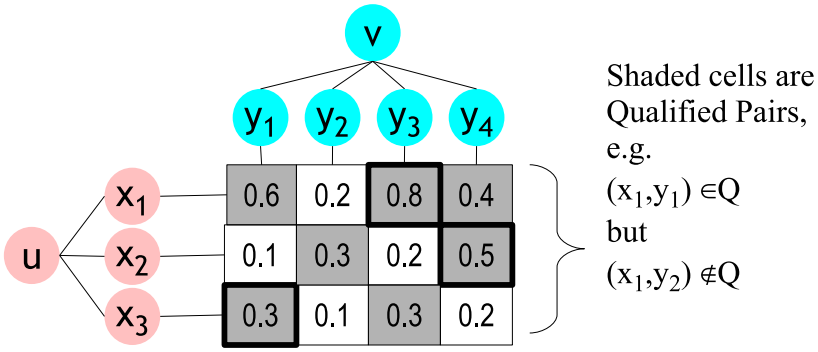


Figure 13: RoleMatch(u,v) Is Constrained to Qualified Neighbors

Symbol and Definition	Description
$Q \subseteq V_0 \times V_1$	set of all qualified global pairings
$Q_i = \{v \mid (v,x) \in Q\}$ *	qualified nodes in V_i
$Q(v) = N(v) \cap Q_i, v \in V_i$	qualified neighbors of v †
$M_Q(u,v) = \{(x,y) \mid x \in Q(u) \wedge y \in Q(v) \wedge (x,y) \in Q\}$	matching between qualified neighborhoods $Q(u)$ and $Q(v)$

Table 7: Qualified Nodes, Neighbors, and Pairings

Qualified Pairings is a subset of all possible node-pairings: $Q \subseteq V_0 \times V_1$. *From this, we derive some related concepts.* Q_i *is the Qualified Node set of* G_i*, composed of the members of* V_i *which appear in* Q. $Q(v)$ *are the Qualified Neighbors of node* $v \in V_i$*, consisting of those neighbors* $N(v)$ *that also appear in* Q.

Table 7 lists our definitions of qualified pairing and several terms related to qualified nodes. Note that a subgraph matching is a special case of a qualified pairing. We can also think of a qualified pairing as a hyperedge matching, as opposed to a bipartite matching. Then, a qualified maximal matching for node-pair (u,v) is constrained to neighbor-pairs that are also within Q.

*Or (x,v), whichever is appropriate for graph G_i.
†$N(v)$ need not be restricted to direct neighbors.

6.3.3 Generalizations for Weighted Edges, Directed Graphs, and Extended Neighborhoods

We briefly note three ways in which RoleMatch can be generalized. First, RoleMatch can easily be extended to match graphs with weighted edges w_{ux}, useful for a host of applications. We simply modify Eq. 29 by replacing the term $R(x, y)$ on the right side with $w(uv, xy)R(x, y)$. From our role-based, neighbor-matching model, we can take two different perspectives on the meaning of edge weights, leading to two definitions of $w(uv, xy)$:
Case 1 (*maxWt*): w_{ux} is the relative importance of x to u's total role. Higher weights should be more influential, so $w_1(uv, xy)$ is simply $w_{ux}w_{yv}$.
Case 2 (*matchedWt*): w_{ux} is a feature of the relationship between u and x. Rather than selecting the highest weights, we want to match similar weights, so the weight factor follows the same generalized Jaccard ratio as overall neighbor matching: $w_2(uv, xy) = \frac{min(w_{ux}, w_{yv})}{max(w_{ux}, w_{yv})}$, which is maximal when $w_{ux} = w_{yv}$.

Second, for directed graphs, we can make two separate matchings, one for in-neighbors and one for out-neighbors, and the total similarity score is the weighted sum of the two. Third, we can extend $N(v)$ to include not just the nodes that are adjacent to v, but rather any node that is within a distance ϵ: $N_\epsilon(v)$. By increasing the ϵ radius, we add more flexibility to the matching options, at the cost of higher computational complexity. An interesting option is to combine $N_\epsilon(v)$ with edge weights: If node w is distance δ from node v, then there is a virtual edge (v, w) with weight $1/\delta$.

6.3.4 RoleMatch Graph Alignment

We now present our complete graph alignment objective function, RoleMatch [80], based on finding the maximal matching of qualified neighborhoods:

Definition 12 RoleMatch Role-based Graph Alignment

Given two graphs G_0 and G_1, determine a qualified subgraph matching M that maximizes

the following objective function:

$$A_M(G_0, G_1) = \frac{\sum_{(u,v)\in M} R_Q(u,v)}{|M|} \left(\frac{|V_0^M| + |V_1^M|}{|V_0| + |V_1|} \right)^{1/\alpha} \tag{30}$$

where

$$R_Q(u,v) = \beta F_{uv} + (1-\beta) \frac{\sum_{(x,y)\in \mathbb{M}_Q(u,v)} R_Q(x,y)}{max(|Q(u)|, |Q(v)|)} \tag{31}$$

Note that M refers to the global alignment, matching nodes between G_0 and G_1, while $\mathbb{M}_Q(u,v)$ refers to a local maximal matching between the (qualified) neighbors of u and v. Also, it is possible for the size of a maximal qualified matching $|\mathbb{M}_Q(u,v)|$ to be smaller than either $|Q(u)|$ or $|Q(v)|$, if Q does not include sufficient pairings between $Q(u)$ and $Q(v)$. If ever $|Q(u)| = 0$ and $|Q(v)| = 0$, then $R_Q(u,v)$ is defined to be βF_{uv}. If node feature similarity function F_{uv} is not provided, then all $F_{uv} = 1$.

Consider how Eq. 30 is affected by the choice of M. If we can identify maximal common subgraphs of G_0 and G_1 (an NP-hard task!), then $S_M(G_0^M, G_1^M) = \frac{\sum_{(i,j)\in M} 1)}{|M|} = 1$. However, the second term attains its maximal value of 1 only if we match all nodes in G_0 and G_1. Thus, the alignment score A_M achieves a maximum value of 1 only if we identify a complete isomorphic matching between the two graphs. Since the term $\frac{|V_0^M|+|V_1^M|}{|V_0|+|V_1|}$ is always between 0 and 1, larger values of α magnify the penalty for having an incomplete matching.

6.4 Computing RoleMatch Alignment

One of the advantages of our recursive formulation is that it can be computed iteratively, starting from any valid set of initial node similarity scores, such as $R_Q^0(x,y) = 1$ for all (x,y). We then apply Eq. 31 iteratively, using the k^{th} iteration's values of R_Q to compute the $(k+1)^{th}$ set.

We are computing both local node similarities and a global alignment, so our iteration cycle has additional steps. After initializing Q and $R_Q^0(u, v)$, each cycle does the following:

1. Iteratively update $R_Q(u, v)$.
2. Compute $A_M(G_0, G_1)$, where the current M is the entire set of qualified pairs Q.
3. Prune Q, eliminating the weakest candidate node-pairs.

Because we are pruning the low-scoring candidates, the average score for the remaining qualified pairings should increase. We stop iterating when the scores stop increasing or when the user does not want to reduce the matching size any more.

6.4.1 Pruning the Global Pairing Q

The objective of pruning the global pairing is to remove node-pairs that do not contribute significantly to the global alignment score. For example, consider the case where it is known that embedded within each of two networks are identical subgraphs. Furthermore, all the other nodes and edges are very different. The pruning operations should remove all the edges and nodes that are not part of the identical subgraphs.

To assess the contribution of node-pair (u, v) to the alignment, we look at each situation where it was an eligible candidate for a local matching. Then, we can rank the contributions and prune away the node-pairs that make the smallest contributions. For each such case, we count whether it was selected and how much benefit it contributed to the total score. For example, if u has 3 neighbors and v has 4, then there are up to 12 candidate node-pairs, but the local matching will select only 3. Nine will be unused. If a node-pair is never selected for any local matching, then clearly it can be eliminated from Q. If a node-pair is always selected whenever it is eligible, then we should retain it. When the voting is not so clear-cut, we need heuristics to guide us. We present three possible heuristics for pruning node-pairs.

- $R_Q(u, v)$ global rank

  Order all the $R_Q(u, v)$ scores and remove node-pairs with the lowest scores.

- $R_Q(u, v) < \theta$

  Remove any node-pair with a score below an absolute threshold θ.

- Number of votes for $R_Q(u, v)$

  Prune based on how many times (u, v) was selected, relative to the maximum possible number of votes and the expected number of votes. A simpler version is to just prune those that were not used at all.

Even more sophisticated heuristics are possible, such as estimating the marginal benefit of a node-pair: how much does it increase the scores compared to the case where it was not qualified? We favor the simplest method that provides effective pruning. We discuss our empirical results in the Experiments section.

6.4.2 Basic RoleMatch Algorithm

Algorithm 2 shows our pruning-based alignment algorithm. While pruning is optional for local similarity calculation, it speeds up the overall performance. The initialization (line 2) can either be very basic, or it can employ pre-pruning, described in the next section. After initialization, we begin iterating. In each iteration, we update local similarity (lines 6-9), compute a bipartite global matching (line 10),compute a global alignment score (line 11), and then prune Q (lines 12-14). The global matching follows a simple greedy design: Start with the highest ranked node-pair. Continue by selecting the highest-ranked pair of unmatched nodes that is adjacent to any node-pair of the current alignment component. Pair (x, y) is adjacent to (u, v) is edge $(x, u) \in E_0$ and $(y, v) \in E_1$. When there are no more adjacent node-pairs, start a new component, until the alignment score A is no longer increasing, or until there are no more unmatched nodes.

Algorithm 2 RoleMatch($G_0, G_1, F, \alpha, \beta, \rho, \theta, z$)

1: Initialize $k = 0, A^0 = 0, Q_0 = V_0, Q_1 = V_1$
2: $R_Q^0 = InitializeQualifiedPairs(G_0, G_1, F, z, \theta)$
3: **repeat**
4: $k = k + 1$
5: Reset $Used(*, *) = false$
6: **for all** $u \in Q_0, v \in Q_1$ **do**
7: $R_Q^k(u, v) = (1 - \beta)\frac{\sum_{(x,y)\in \mathbb{M}_Q(u,v)} R_Q^{k-1}(x,y)}{max(|Q(u)|,|Q(v)|)} + \beta F_{uv}$
8: **for each** $(x, y) \in \mathbb{M}_Q(u, v)$: $Used(x, y) = true$
9: **end for**
10: Alignment $M = GreedyConnectedMatching(Q)$
11: Score $A_M^k = \frac{\sum_{(u,v)\in M} R_Q^k(u,v)}{|M|} \left(\frac{|V_0^M|+|V_1^M|}{|V_0|+|V_1|}\right)^{1/\alpha}$;
12: Prune Q: Remove (u, v) if $Used(u, v) = false$ AND
13: 1. Either $R_Q^k(u, v) < \theta$
14: 2. Or $R_Q^k(u, b)$ is in the lowest ρ % of values.
15: **until** $A^k < A^{k-1}$
16: **return** M^{k-1} and A^{k-1}

6.4.3 Scalable RoleMatch

If the graphs have only a few thousands of nodes, then the memory and time requirements are reasonable for fully realized matrices. We can initialize each node-pair (Step 2 of Algorithm 2) with this simple formula:

Case 1: Initialize small graphs

$$R_Q^0(u, v) = (1 - \beta) + \beta F_{uv} \tag{32}$$

However, $O(n^2)$ matrices are too expensive for large graphs, so we apply the following initial pre-pruning. We take advantage of the fact that there is no need to store initial local similarity value $R_Q^0(u, v)$ because it can be computed on the fly from $F(u, v)$. Thus, we can skip ahead to the next iteration.

Case 2: Initialize large graphs with F_{uv}

$$R_Q^1(u,v) = (1-\beta)\frac{\sum_{(x,y)\in\mathbb{M}^0(u,v)} R_Q^0(x,y)}{max(d_u,d_v)} + \beta F_{uv}$$
$$= (1-\beta)\frac{\left((1-\beta)|\mathbb{M}^0| + \beta \sum_{(x,y)\in\mathbb{M}^0(u,v)} F_{xy}\right)}{max(d_u,d_v)} + \beta F_{uv} \quad (33)$$

where $|\mathbb{M}^0| = min(d_u, d_v)$.

However, in the event that there are no initial feature values, then local similarity depends only on matching node degrees of neighbors. R^0 degenerates to all 1, and $R^1(u,v) = \frac{min(d_u,d_v)}{max(d_u,d_v)}$. In this case, we pre-compute $R^2(u,v)$ as follows:

Case 3: Initialize large graphs without F_{uv}

$$R^2(u,v) = (1-\beta)R^1(u,v)\left(\sum_{(x,y)\in\mathbb{M}^0(u,v)} (1-\beta)R^1(x,y) + \beta\right) + \beta F_{uv} \quad (34)$$

To reduce the memory requirements, we use a hash table of limited size to store only the highest $R_Q(u,v)$ values. Hash tables are used for Q and $Used$ as well. To select which node-pairs will be stored in the hash tables, we filter out node-pairings that cannot have a high similarity score:

Lemma 4 Local Similarity Upper Bound: *An upper bound for $R_Q(u,v)$ is given by*

$$R_Q(u,v) \leq (1-\beta)\frac{min(d_u,d_v)}{max(d_u,d_v)} + \beta F_{uv}$$

Proof: Since this formula depends only on node degree, we can efficiently perform the filtering using degree-pairs instead of node-pairs. Referring to the definition of local similarity, we observe that a matching $M(u,v)$ is the sum of $min(d_u,d_v)$ similarity scores, each of which has a maximum value of 1. □

Algorithm 3 InitializeQualifiedPairs(G_0, G_1, F, θ, z)

1: Bin nodes by degree: $D_i(r) = \{v \in G_i | d_v = r\}$
2: {Select top z fraction of degree-qualified node-pairs}
3: Set of initial scores $R_Q^0 = \emptyset$
4: **for all** $u \in V_0$ **do**
5: Clear priorityQueue P, which sorts by R_Q values.
6: **for all** degree values $d \in D_1$ **do**
7: **if** $(1-\beta)\frac{min(d_u,d)}{max(d_u,d)} + \beta \geq \theta$ **then**
8: **for all** $v \in D_1(d)$ **do**
9: Compute $R_Q(u,v)$ using Eq. (33) or (34)
10: Add $R_Q(u,v)$ to P; Keep the top $z|V_1|$ values
11: **end for**
12: **end if**
13: **end for**
14: $R_Q^0 = R_Q^0 \cup P$
15: **end for**
16: **return** R_Q^0

Employing this lemma, we develop Algorithm 3 to initialize Q, which is Line 2 of the full alignment algorithm. The two parameters θ and z control how much pre-pruning is performed. Both parameters should be in the range 0 to 1. Any node-pair whose upper bound similarity scores is below θ is not included (line 7). Of the remaining nodes, we select the top $z|V|$ partners for each node (lines 9-10). Hence, to a first approximation, z is the size scaling factor between the Q hash table and a complete $|V_0||V_1|$ matrix of local similarity scores.

There is one important difference here between Iceberg RoleSim's pre-pruning and RoleMatch's pre-pruning. In RoleSim, we still need some score for node-pairs that are not stored, to do complete neighborhood matching, so we compute an on-the-fly value between the lower bound and the upper bound. For RoleMatch, we reject node-pairs are that not stored. Their removal reduces the size of the neighborhoods. No estimate is needed.

6.4.4 Computational Complexity

In Algorithm 2 (RoleMatch iterations), the most expensive step is Step 7, computing a maximal matching of qualified neighbors, for each member of Q. Our experiments show that a greedy approximate matching is sufficient, which has average complexity $O(\hat{d}^2 \log \hat{d}^2)$ per node-pair, where $\hat{d}$ is the average qualified degree of a node [3]. Thus, the overall complexity is $O(k|Q|\hat{d}^2 \log \hat{d}^2)$. The number of iterations k is typically no more than 5. The upper bound for $|Q|$ is $n_0 n_1$, though pre-pruning and iterative pruning reduces this.

Algorithm 3 (InitializeQualifiedPairs) performs a one-time arithmetic test on at most $\binom{n}{2}$ node-pairs. If we pre-sort nodes by degree, we only have to perform this per-block rather than per-node. The parameter θ determines what portion of the full node-pair set requires more computation. The scalability effect of θ depends on graph characteristics, but for power law graphs, a linear increase in θ causes an exponential decrease in Q. The memory cost is linear with an adjustable constant factor: $z \max(n_0, n_1)$.

6.5 Experimental Evaluation

We now evaluate RoleMatch's graph matching performance in a variety of situations and applications. We wish to answer the follow questions: (1) How well does RoleMatch perform subgraph matching, for both exact (isomorphic) and inexact matches? How much noise or dissimilarity can it tolerate? (2) Can RoleMatch outperform existing approaches for global PPI network alignment, where the graphs are distantly related, but *a priori* node label similarity information is available as clues? (3) Can we deanonymize nodes in an unlabeled network, by comparing it to a related network with labels? Except where noted, the algorithms are implemented in C++ and run on a Linux server with a 3.2GHz Xeon dual-core processor and 16GB of RAM.

[3] An even faster greedy matching algorithm is available, with $O(\hat{d}^2)$ time [100]

Test	Method for Creating Second Graph from First Graph
Exact graph matching	Shuffle node names (for all tests)
Inexact graph matching	Reroute randomly selected edges
Exact subgraph matching	Remove randomly selected edges
Inexact subgraph matching	Remove and reroute randomly selected edges

Table 8: Constructing Graphs for Comparison Tests

6.5.1 Matching Synthetic Graphs

We use synthetic graphs to answer basic matching quality questions: How well can we match isomorphic graphs? How well can we match a subgraph to its supergraph? As increasing amounts of topological disturbance are introduced into one of our graphs, how does this affect our matching performance?

Graphs: We generated two types of random graphs: power law graphs and Erdős-Rényi $G(n, p)$ graphs. We varied the number of nodes from 100 up to 30 000 and the average node degree from 2 to 8. For each configuration, we generated 3 random instances. Each test of course requires two graphs. The two could be equally sized or not, and they might match exactly or not. Table 8 outlines how we constructed each pair of graphs for these for cases.

Isomorphic Graph Matching

Though in the worst case graph isomorphism discovery is non-polynomial, many heuristic algorithms do a reasonable job for realistic graphs. We tested RoleMatch(RM) and RoleMatch with prepruning (RMpp) against the following algorithms: Umeyama's eigendecomposition (UM) [101], IsoRank(IR) [97] based on local similarities, and several based on solving relaxations of linear programming equations: PATH [94], MP, and GA(both

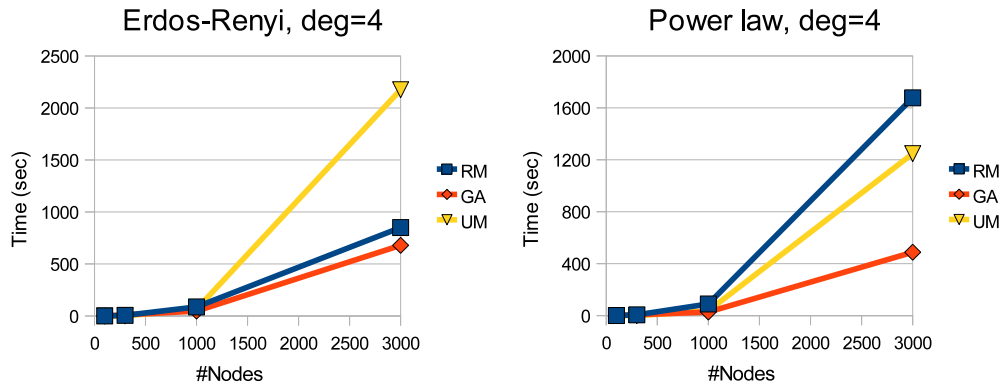


Figure 14: Time vs. Graph Size, deg=4

from [95]). We executed these using the GraphM simulation environment [4]. We discovered that three of the algorithms, PATH, IR, and MP, were either much slower or much less accurate than the other four, so we do not report their detailed results in this section.

In our first trials, we did not shuffle the node IDs between the two graphs. In this situation, every algorithm except IR was able to match 100% of the nodes most of the time, occasionally dipping as low at 97%. IsoRank is not designed for problems where there is no initial node similarity knowledge, so its scores were poor. When we shuffled the nodes, however, we were surprised to discover that GA's performance dropped to near zero. GA uses a heuristic method of finding a "good" starting point, so it may have a bias towards the identify permutation. Apparently it has trouble if it does not find a good starting point.

RoleMatch (RM) without prepruning can be slow, due to maximal matching for every node-pairs neighbor. Other algorithms are more scalable, but as later tests show, they have issues with the quality of the matching. Figure 14 compares the execution times of RM, GA, and UM, for both types of random graphs, with average node degree 4, up to 3 000 nodes per graph. For Erdős-Rényi graphs, RM is roughly comparable to GA and better than UM. However, for power law graphs, RM is the slowest and with the highest rate

[4] *http://cbio.ensmp.fr/graphm/*

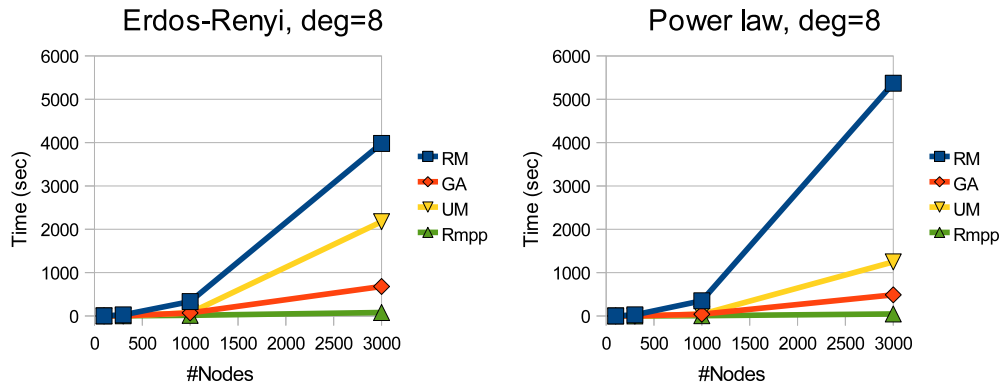


Figure 15: Time vs. Graph Size, deg=8, including RMpp

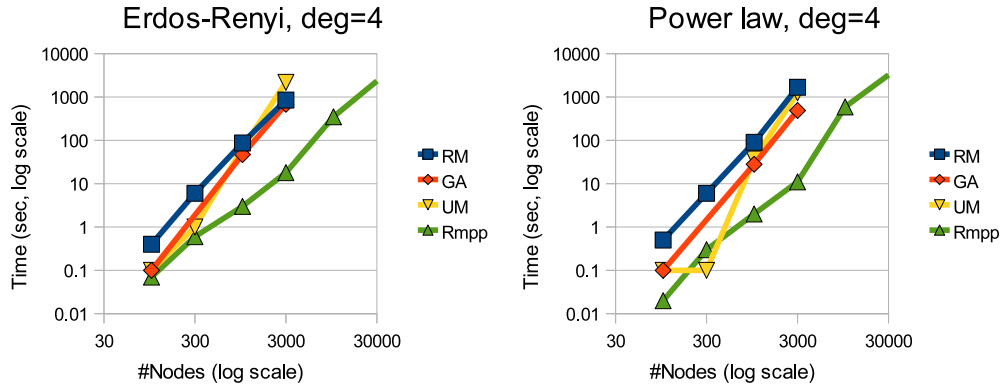


Figure 16: Time vs. Graph Size, deg=4, log-log Scale, Extended Graph Sizes

of growth. However, the situation changes radically when RoleMatch with pre-pruning (RMpp) is included. Figure 15 shows the same type of tests, except that the node degree is increased to 8. RM's speed is getting even worse as it is sensitive to the number of edges. RMpp, however, is so much faster than the other algorithms that its time seems almost flat at this scale. To test the limits of RMpp's performance, we increased the graph size up to 30 000 nodes. None of the other algorithms could handle graphs of even 10 000 nodes, probably due to excessive memory usage. We now display the execution time on log-log charts (Figure 16). RMpp is about one order of magnitude faster than GA, the next

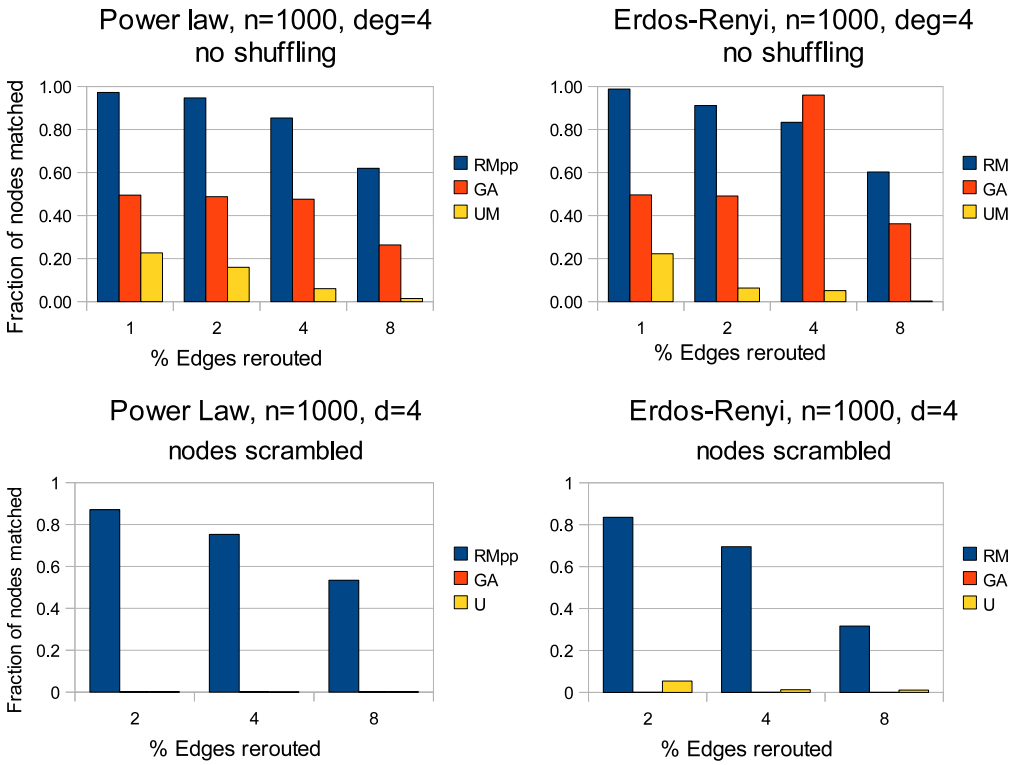


Figure 17: Inexact Graph Matching

fastest algorithm, completing the matching of graphs with 30 000 nodes and 120 000 edges in about 3 hours. For this experiment, comparing isomorphic graphs, RMpp's matching quality was virtually perfect.

Inexact Graph Matching

Now we reroute up to 8% of the edges of one graph so that the two are no longer identical. Each edge change will affect two nodes plus the local similarity of several neighbors, so the maximum possible matching performance is approximately $(100-2\delta)\%$. RoleMatch is still able to match most of the unaltered nodes (Figure 17). GA does quite well when we do not shuffle nodes (upper charts), but it ceases to work when nodes are shuffled (lower charts).

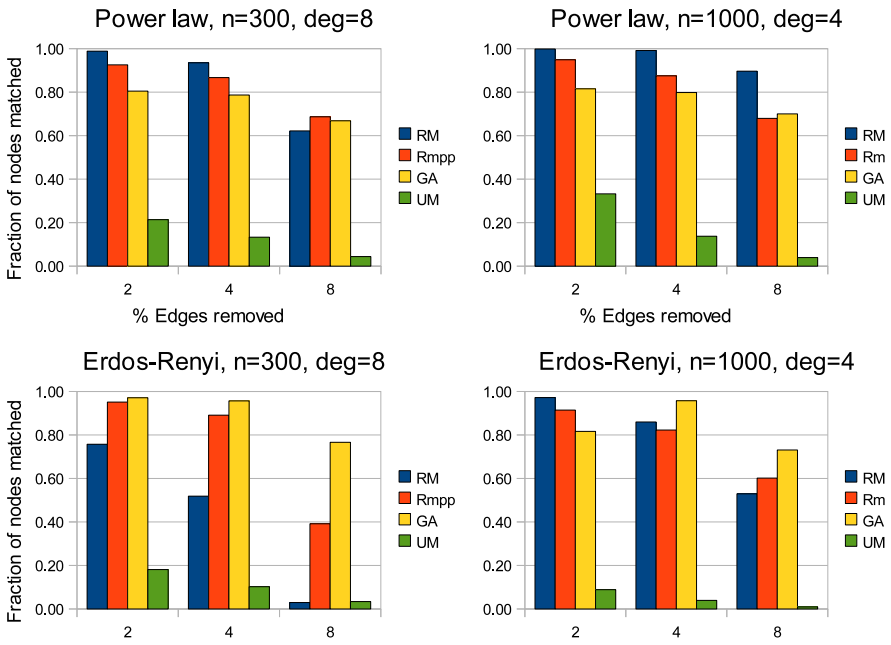


Figure 18: Subgraph Matching

Subgraph Matching

To create graphs for this test, we begin with two identical graphs and then randomly removed a percentage of nodes and incident edges from the second graph so that what remains is an induced subgraph of the first one. We ran 3 trials for each configuration, with different randomly generated subgraphs for each trial. A score of 1.0 means that all of the subgraph's nodes were correctly matched to the larger graph. Figure 18 shows the average matching performance for each algorithm. Looking at average performance only, there is no clear leader among RM, RMpp, and GA. However, RM has much lower variance. We calculated standard deviation for each configuration. RM's median standard deviation $\sigma = 0.03$, while GA's median $\sigma = 0.28$.

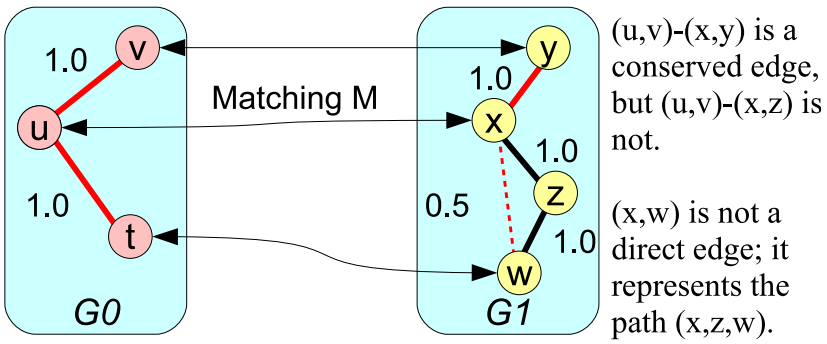


Figure 19: Conserved Edges and Weights

6.5.2 Detecting Conserved Protein Interactions

We now examine matching performance with the first of two real-world datasets. Given two PPI networks, can RoleMatch find the best 1-1 alignment that also conserves the most interactions within the networks? Two pairs (u, x) and (v, y) in matching M describe a *conserved interaction* if and only if there exist edges $(u, x) \in G_0$ and $(v, y) \in G_1$ (See Figure 19).

We start with a baseline experiment that replicates a test in Zaslavskiy [95], which in turn follows [74]. This provides a benchmark comparison between RoleMatch and four other well-regarded alignment algorithms. Then, using more extensive PPI data, we investigate whether RoleMatch can discover more conserved interactions than other approaches. All of the input data and the executable programs for algorithms other than RoleMatch were downloaded from `cbio.ensmp.fr/proj/graphm_ppi/`.

Our input data include two small PPI subgraphs for two species, fruitfly (356 nodes) and yeast (256 nodes). These subgraphs are chosen because prior knowledge shows that these regions contain most of the conserved interactions. An interaction path may have evolved by having a node inserted into or deleted from the path. We want our algorithms to consider these off-1 matches, so we enhance the graphs: for every pair of nodes that are

exactly distance=2 apart, we add a new edge, with 1/2 the weight of a direct edge. The product of the two edge weights is the quality of the match. Figure 19 shows an example: The quality of off-1 match $(u,t)-(x,w)$ is $w(x,t)w(x,w) = 1.0 * 0.5 = 0.5$.

Our input also includes a bipartite clustering of the proteins, provided by the InParanoid database[5]. Each cluster contains at least one fly protein and at least one yeast protein. Clusters that contain more than the minimum number are called ambiguous, because it is unknown which fly protein is the best match for which yeast protein. We use these clusters as a prior constraint on our alignment: each matched pair should be between members of the same cluster.

Let X and Y be the adjacency matrices for the fly and yeast networks, respectively, where Y is extended with dummy nodes so that it is equal in size to X. Let C be the cluster binary indicator matrix: $C_{xy} = 1$ iff fly protein x clusters with yeast protein y. Let M be the binary indicator matrix for the final matching result. Following [95], we score an alignment as the sum of the weights of all the conserved interactions, computable as $Q(M) = \frac{1}{2}tr(X^T M Y M^T)$.

We ran RoleMatch with $\beta = 0.75$ and with two weight options: ignoring weights (RM) and with the *maxWt* option (RMx) (described in Section 6.3.3). We compared to several of the algorithms mentioned in the synthetic tests: MP, IR, PATH, and GA. The C matrix is used as the feature matrix for RoleMatch, the initial node similarity state for IsoRank, and as the constraint matrix for the others.

We also used this opportunity to experiment with RoleMatch's pruning options. We discovered that pruning node-pairs which both have the lowest ranked similarity scores *and* also were not used for matching was the most effective option. Using a fixed threshold θ is problematic, because it is difficult to select θ.

Table 9 shows our results. We scored each matching three ways: counting only direct

[5] `http://inparanoid.sbc.su.se/`

Algorithm	MP	Path	GA	IR	RM	RMx
direct matches only	28	28	28	25	26	26
Permit one graph dist=2 edge	124	124	124	114	122	111
Permit both graphs dist=2 edges	243	245	238	245	412	381
time (secs)	1-2	80-100	1-2	1-2	3	3

Table 9: Conserved Interaction Scores, for Small PPI

Algorithm	Path	GA	IR	RM
Permit both graphs dist=2 paths	939	942	900	1642
time (secs)	1542	372	31	241

Table 10: Conserved Interaction Scores, for Larger PPI

interactions, permitting a matched interaction to have one distance-2 edge, and permitting a matched interaction to have two distance-2 edges. In the first two cases, Path and GA obtain the best results, while IsoRank and RoleMatch are both within a few percent of this. For the third case, RoleMatch surpasses all the other methods. RoleMatch specifically tries to find the best match of all neighbors to all neighbors. Surprisingly, RMx does not do as well as RM. This may

We repeated the experiment using a larger subset of the full PPI networks: 984 fly nodes and 736 yeast nodes. We again modified the original graphs to include distance-2 paths; however, we weighted all edges the same. This graph is too large for the MP algorithm. Table 10 shows that RoleMatch finds significantly more conserved edges. It is also faster than most of the other approaches.

6.5.3 Matching Time-Evolved Coauthor Networks

In this test, we construct overlapping coauthor networks to test subgraph alignment and node deanonymization. We use the entire DBLP database of academic papers [102] to generate a database of coauthor network links, each link labeled by year and by publishing venue. By selecting for certain years or venues, we can create many possible coauthor networks.

Algorithm	IR	GA	UM	QCV	RMpp	RM	Truth
Nodes matched	8	4	16	305	373	408	712
Edges matched	87	1152	281	1453	1286	1584	1796
time (secs)	7	63	47	125	8	45	n/a

Table 11: Overlapping ICDM Coauthor Networks

We created two overlapping sets of coauthor edges: ICDM papers published 2002 to 2010 and those published 2003 to 2011. After reducing the edge sets to their largest connected components, we were left with networks containing 820 nodes and 972 nodes, respectively. The two networks share 712 nodes and 1754 edges, which represent the ideal matching result. Using unlabeled and unweighted networks, we performed alignment using RoleMatch and compared to GA, UM, IR, and QCV [94], as implemented in the GraphM package.

Table 11 compares the performance of the algorithms.

RM has the best alignment quality for nodes and edges. Node matches are counted using our knowledge of the hidden node labels (author name). RM and RMpp are the only algorithms to correctly identify more than half of the unlabeled nodes. Edge matches are counted using graph permutation: if an edge exists in one graph, does the alignment map this to some edge in the other graph? This measure is based purely on local topological equivalence and does not reply on any ground truth matching. Interestingly, there is a large discrepancy between GA's edge matching and node matching performance. GA's objective function is based on edge matching. When feature similarity knowledge is absent, it has no measure of node-based local similarity. The pre-pruned version essentially tied for fastest method, while having the second best match quality.

We attempted another pair of coauthor graphs with approximately 10K nodes and 36K edges per graph with 7699 nodes in common. We desired to perform a comparison, but none of the other algorithms were able to process graphs of this size. RoleMatch completed

the alignment in 18 minutes. It correctly matched 3849 (50%) of the nodes and aligned 26 435 edges.

CHAPTER 7

Conclusion

This dissertation has presented how a social role model can be productively applied to measure similarity in network structures. We first considered the meanings of structural equivalence and similarity in order to devise an axiomatic definition of role similarity. These axioms offer a standard means for validating any future proposals for a node similarity measure. This axiomatic approach may prove useful for developing and validating solutions to other related tasks.

We then developed RoleSim, the first real-valued role similarity measure that confirms automorphic equivalence. RoleSim is based on a recursive definition of automorphic equivalence and a generalized Jaccard coefficient which measures the distance away from perfect equivalence. RoleSim can be computed with a simple iterative algorithm that is guaranteed to converge. Our experimental tests demonstrate RoleSim's correctness and usefulness on real world data, opening up exciting possibilities for scientific and business applications. At the same time, we see that other well-known measures, while suitable for other tasks, are not suitable for role similarity. We also devised a more scalable version of RoleSim, which applies theoretical upper and lower bounds on similarity values to compute estimated values of less critical node-pairs. In doing so, the memory needs can be greatly reduced.

RoleSim's recursive node similarity measure extends naturally to global graph matching or network alignment. Our formulation, RoleMatch, admits an upper-bound limit and look-ahead calculation of similarity scores, enabling substantial pre-pruning of matching candidates. This in turn provides scalable computation time and nearly linear memory cost. Experiments demonstrate that is it effective for a variety of different graph types and applications, from graphs that are nearly identical, to those that are substantially different but

evolutionarily related.

The current work lays a promising foundation for future research in role-based analysis of networks. To further improve the scalability, both RoleSim and RoleMatch are excellent candidates for parallel computation. One could also investigate how statistical sampling or machine learning could obtain estimated values at lower cost than full calculations. Another improvement would be to extend RoleMatch to align more than two graphs, while maintaining efficiency.

Looking further afield, role-based network analysis suggests several exciting questions: Do certain roles or multi-role structures determine network growth and evolution? How are roles likely to change over time? Can social network managers encourage and engineer people to fulfill certain roles? What is a practical implementation of role-based product recommendations? Some networks have a backbone through which support the bulk of intranetwork communication. How does backbone structure relate to roles? These questions, and others yet to be formed, all build upon what this dissertation has established: role analysis offers not just a mathematical tool but a socially-based interpretation of network structure and function.

BIBLIOGRAPHY

[1] J. Donne, "Meditation xvii," in *Luminarium: Anthology of English Literature*, Accessed July 4 2012. [Online]. Available: http://www.luminarium.org/sevenlit/donne/meditation17.php/

[2] A. Miller, "Untangling the social web," *The Economist*, Sep 2 2010. [Online]. Available: http://www.economist.com/node/16910031/

[3] R. E. Ulanowicz, "Quantitative methods for ecological network analysis," *Comput. Biol. Chem.*, vol. 28, no. 5-6, pp. 321–339, Dec. 2004. [Online]. Available: http://dx.doi.org/10.1016/j.compbiolchem.2004.09.001

[4] B. Schwikowski, P. Uetz, and S. Fields, "A network of protein-protein interactions in yeast," *Nature biotechnology*, vol. 18, no. 12, pp. 1257–1261, 2000.

[5] A. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[6] K. I. Calvert, M. B. Doar, and E. W. Zegura, "Modeling internet topology," *IEEE Comm. Magazine*, vol. 35, no. 6, pp. 160–163, 1997.

[7] V. Batagelj, "Efficient algorithms for citation network analysis," *Computing Research Repository (CoRR)*, vol. cs.DL/0309023, 2003.

[8] P. Forster, A. Toth, and H.-J. Bandelt, "Evolutionary network analysis of word lists: visualising the relationships between alpine romance languages," *Journal of Quantitative Linguistics*, vol. 5, no. 3, pp. 174–187, 1998.

[9] M. Gladwell, *The tipping point: How little things can make a big difference*. Little, Brown and Company, 2000.

[10] L. Wiskott, J.-M. Fellous, N. Kuiger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775–779, 1997.

[11] O. A. Oeser and F. Harary, "A mathematical model for structural role theory. part i." *Human Relations*, vol. 15, no. 2, pp. 89–109, 1962.

[12] ——, "A mathematical model for structural role theory. part ii." *Human Relations*, vol. 17, no. 1, pp. 3–17, 1964.

[13] F. P. Lorrain and H. C. White, "Structural equivalence of individuals in networks," *J. Math. Sociology*, vol. 1, pp. 49–80, 1971.

[14] L. D. Sailer, "Structure equivalence: meaning and definition, computation and application," *Social Networks*, vol. 1, pp. 73–80, 1978.

[15] D. R. White and K. P. Reitz, "Graph and semigroup homomorphisms on networks of relations," *Social Networks*, vol. 5, pp. 193–234, 1983.

[16] S. P. Borgatti and M. G. Everett, "Two algorithms for computing regular equivalence," *Social Networks*, vol. 15, pp. 361–376, 1993.

[17] M. G. Everett and S. P. Borgatti, "Exact colorations of graphs and digraphs," *Social Networks*, vol. 18, pp. 319–331, 1996.

[18] S. Fortin, "The graph isomorphism problem," Dept. Computer Science, Univ. of Alberta, Edmonton, Alberta, Canada, Tech. Rep. TR 96-20, Jul. 1996.

[19] "Database of interacting proteins." [Online]. Available: http://dip.doe-mbi.ucla.edu/dip/Stat.cgi

[20] S. P. Borgatti and M. G. Everett, "Notions of position in social network analysis," *Sociological Methodology*, vol. 22, pp. 1–35, 1992.

[21] C. Godsil and G. Royle, *Algebraic Graph Theory*. Springer-Verlag, 2001.

[22] D. M. Cvetkovíc, M. Doob, and H. Sachs, *Spectra of Graphs: Theory and Applications, 3rd Revised and Enlarged Edition*. Wiley, 1998.

[23] R. Read and D. Corneil, "The graph isomorphism disease," *J. Graph Theory*, vol. 1, pp. 339–363, 1977.

[24] B. McKay, "Practical graph isomorphism," *Congressus Numerantium*, vol. 30, pp. 45–87, 1981.

[25] M. Marx and M. Masuch, "Regular equivalence and dynamic logic," *Social Networks*, vol. 25, no. 1, pp. 51–65, 2003.

[26] V. Batagelj, P. Doreian, and A. Ferligoj, "An optimizational approach to regular equivalence," *Social Networks*, vol. 14, pp. 121–135, 1992.

[27] P. Doreian, V. Batagelj, and A. Ferligoj, *Generalized blockmodeling*. Cambridge Univ Press, 2005, vol. 25.

[28] V. E. Lee, N. Ruan, R. Jin, and C. Aggarwal, *Managing and Mining Graph Data*. Springer, 2010, ch. 10: A Survey of Algorithms for Dense Subgraph Discovery.

[29] J. D. Noh and H. Rieger, "Random walks on complex networks," *Phys. Rev. Lett.*, vol. 92, p. 118701, Mar 2004.

[30] K. Stephenson and M. Zelen, "Rethinking centrality: Methods and examples," *Social Networks*, vol. 11, no. 1, pp. 1–37, 1989.

[31] L. C. Freeman, “A set of measures of centrality based on betweenness,” *Sociometry*, pp. 35–41, 1977.

[32] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank Citation Ranking: Bringing Order to the Web,” Stanford InfoLab, Tech. Rep. 1999-66, 1999. [Online]. Available: http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf

[33] T. Milenković and N. Pržulj, “Uncovering biological network function via graphlet degree signatures,” *Cancer Informatics*, vol. 6, pp. 257–273, 2008.

[34] M. M. Kessler, “Bibliographic coupling between scientific papers,” *American Documentation*, vol. 14, no. 1, pp. 10–25, 1963.

[35] H. Small, “Co-citation in the scientific literature: A new measure of the relationship between two documents,” *J. Amer. Soc. Information Sci.*, vol. 24, pp. 265–269, 1973.

[36] T. T. Tanimoto, “An elementary mathematical theory of classification and prediction,” *IBM Taxonomy Application M. and A.6*, vol. 3, Nov 1958.

[37] M. Schultz and M. Liberman, “Topic detection and tracking using idf-weighted cosine coefficient,” in *Proc.e DARPA Broadcast News Workshop*, 1999, pp. 189–192.

[38] G. Jeh and J. Widom, “Simrank: a measure of structural-context similarity,” in *Proc. 8th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2002, pp. 538–543.

[39] Z. Lin, M. R. Lyu, and I. King, “Extending link-based algorithms for similar web pages with neighborhood structure,” in *Proc. IEEE/WIC/ACM Int’l Conf. Web Intelligence*, 2007, pp. 263–266. [Online]. Available: http://www.cse.cuhk.edu.hk/~king/PUB/WI2007_Lin.pdf

[40] P. Zhao, J. Han, and Y. Sun, “P-rank: a comprehensive structural similarity measure over information networks,” in *Proc. 18th ACM conf. Inform. and knowledge manage. (CIKM)*, 2009, pp. 553–562.

[41] D. Lizorkin, P. Velikhov, M. Grinev, and D. Turdakov, “Accuracy estimate and optimization techniques for simrank computation,” *Proc. VLDB Endow.*, vol. 1, pp. 422–433, 2008.

[42] X. Jia, Y. Cai, H. Liu, J. He, and X. Du, “Calculating similarity efficiently in a small world,” in *Proc. 5th Int. Conf. Advanced Data Mining and Applications (ADMA)*, 2009, pp. 175–187.

[43] Y. Cai, G. Cong, X. Jia, H. Liu, J. He, J. Lu, and X. Du, “Efficient algorithm for computing link-based similarity in real world networks,” in *Ninth IEEE Int. Conf. Data Mining (ICDM)*, 2009, pp. 734–739.

[44] P. Li, Y. Cai, H. Liu, J. He, and X. Du, "Exploiting the block structure of link graph for efficient similarity computation," in *Proc. 13th Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining (PAKDD)*, 2009, pp. 389–400.

[45] I. Antonellis, H. Garcia-Molina, and C.-C. Chang, "Simrank++: query rewriting through link analysis of the clickgraph," in *Proc. VLDB Endow.*, vol. 1, no. 1, 2008, pp. 408–421.

[46] D. Fogaras and B. Rácz, "Scaling link-based similarity search," in *Proc. 14th Int. Conf. World Wide Web (WWW)*, 2005, pp. 641–650.

[47] Z. Lin, M. R. Lyu, and I. King, "Matchsim: a novel neighbor-based similarity measure with maximum neighborhood matching," in *Proc. 18th ACM conf. Inform. and knowledge manage. (CIKM)*, 2009, pp. 1613–1616.

[48] Z. Lin, I. King, and M. R. Lyu, "Pagesim: A novel link-based similarity measure for the world wide web," in *Proc. IEEE/WIC/ACM Int'l Conf. Web Intelligence*, 2006, pp. 687–693.

[49] E. A. Leicht, P. Holme, and M. E. J. Newman, "Vertex similarity in networks," *Phys. Rev. E*, vol. 73, p. 026120, Feb 2005.

[50] S. Wasserman and K. Faust, *Social network analysis: methods and applications*. Cambridge University Press, 1994.

[51] J. J. Luczkovich, S. P. Borgatti, J. Johnson, and M. G. Everett, "Defining and measuring trophic role similarity in food webs using regular coloration," *J. Theoretical Biology*, vol. 220, no. 3, pp. 303–321, 2003.

[52] E. M. Hafner-Burton, M. Kahler, and A. H. Montgomery, "Network analysis for international relations," *International Organization*, vol. 63, no. 03, pp. 559–592, 2009.

[53] N. Dragan, M. L. Collard, and J. I. Maletic, "Using method stereotype distribution as a signature descriptor for software systems," in *IEEE Int. Conf. Software Maintenance (ICSM)*, 2009, pp. 567–570.

[54] P. Holme and M. Huss, "Role-similarity based functional prediction in networked systems: application to the yeast proteome," *J. R. Soc. Interface*, vol. 2, no. 4, pp. 327–33, 2005.

[55] M. K. Sparrow, "A linear algorithm for computing automorphic equivalence classes: the numerical signatures approach," *Social Networks*, vol. 15, no. 2, pp. 151–170, 1993.

[56] B. D. MacArthur, R. J. Sánchez-García, and J. W. Anderson, "Note: Symmetry in complex networks," *J. Discrete Applied Math.*, vol. 156, no. 18, pp. 3525–3531, 2008.

[57] W. Xi, E. A. Fox, W. Fan, B. Zhang, Z. Chen, J. Yan, and D. Zhuang, "Simfusion: measuring similarity using unified relationship matrix," in *Proc. 28th Int. ACM SIG Conf. Research and Develop. in Inform. Retrieval (SIGIR)*, 2005, pp. 130–137.

[58] X. Yin, J. Han, and P. S. Yu, "Linkclus: efficient clustering via heterogeneous semantic links," in *Proc. 32nd Int. Conf. Very Large Data Bases (VLDB)*, 2006, pp. 427–438.

[59] R. Jin, V. E. Lee, and H. Hong, "Axiomatic ranking of network role similarity," http://arXiv.org, Tech. Rep. arXiv:1102.3937, 2011.

[60] G. Melançon and A. Sallaberry, "Edge metrics for visual graph analytics: A comparative study," in *Proc. 12th Int. Conf. Inform. Visual.*, 2008, pp. 610–615.

[61] H. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[62] D. Avis, "A survey of heuristics for the weighted matching problem," *Network*, vol. 13, pp. 475–493, 1983.

[63] Y. J. Wang and G. Y. Wong, "Stochastic blockmodels for directed graphs," *J. American Statistical Assoc.*, vol. 82, no. 397, pp. 8–19, 1987.

[64] H. White, S. Boorman, and R. Breiger, "Social structure from multiple networks. i: Blockmodels of roles and positions," *Am. J. Sociology*, vol. 81, pp. 730–780, 1976.

[65] J. Tang, J. Zhang, L. Yao, and J. Li, "Extraction and mining of an academic social network," in *Proc. 17th Int. conf. World Wide Web (WWW)*, 2008, pp. 1193–1194.

[66] S. L. Tauro, G. Siganos, C. Palmer, and M. Faloutsos, "A simple conceptual model for the internet topology," in *Proc. IEEE Global Telecomm. Conf.*, 2001, pp. 1667–1671.

[67] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir, "A model of internet topology using k-shell decomposition," *Proc. Nat.Academy of Sci. (PNAS)*, vol. 104, no. 27, pp. 11 150–11 154, July 2007.

[68] S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. 3rd ACM Symp.Theory of Computing (STOC)*, ser. STOC '71. New York, NY, USA: ACM, 1971, pp. 151–158. [Online]. Available: http://doi.acm.org/10.1145/800157.805047

[69] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD int. conf. Knowledge discovery and data mining (KDD)*, 2003, pp. 137–146.

[70] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," *ACM Trans. Web*, vol. 1, no. 1, May 2007.

[71] D. A. Luke and J. K. Harris, "Network analysis in public health: history, methods, and applications," *Annual Review of Public Health*, vol. 28, pp. 69–93, 2007.

[72] A. Narayanan and V. Shmatikov, "De-anonymizing social networks," in *IEEE Symp. Security and Privacy*, 2009, pp. 173–187.

[73] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, and C. Faloutsos, "It's who you know: Graph mining using recursive structural features," in *Proc. 17th ACM SIGKDD Int. Conf. Knowledge discovery and data mining (KDD)*, 2011, pp. 663–671.

[74] S. Bandyopadhyay, R. Sharan, and T. Ideker, "Systematic identification of functional orthologs based on protein network comparison," *Genome research*, vol. 16, no. 3, pp. 428–435, 2006.

[75] M. T. Dittrich, G. W. Klau, A. Rosenwald, T. Dandekar, and T. Müller, "Identifying functional modules in protein–protein interaction networks: an integrated exact approach," *Bioinformatics*, vol. 24, no. 13, pp. i223–i231, 2008.

[76] J. Dutkowski and J. Tiuryn, "Identification of functional modules from conserved ancestral protein–protein interactions," *Bioinformatics*, vol. 23, no. 13, pp. i149–i158, 2007.

[77] J. Flannick, A. Novak, C. B. Do, B. S. Srinivasan, and S. Batzoglou, "Automatic parameter learning for multiple network alignment," in *Proc. 12th int. conf. Research in comput. molecular biol. (RECOMB)*, ser. RECOMB'08, 2008, pp. 214–231.

[78] R. Singh, J. Xu, and B. Berger, "Pairwise global alignment of protein interaction networks by matching neighborhood topology," in *Proc. 11th int. conf. Research in comput. molecular biol. (RECOMB)*, ser. RECOMB'07. Springer, 2007, pp. 16–31.

[79] S. Mohammadi and A. Grama, "Chapter 5: Biological network alignment," *Functional Coherence of Molecular Networks in Bioinformatics*, pp. 97–136, 2012.

[80] V. E. Lee, L. Li, T. M. Fox, and R. Jin, "Scalable role-based alignments of large networks," under submission.

[81] R. Jin, V. E. Lee, and H. Hong, "Axiomatic ranking of network role similarity," in *Proc. 17th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*. ©ACM, 2011, pp. 922–930.

[82] J. Köbler, U. Schöning, and J. Torán, *The graph isomorphism problem: its structural complexity*, ser. Progress in Theoretical Computer Science. Birkhauser, 1993.

[83] H. Bunke, P. Foggia, C. Guidobaldi, C. Sansone, and M. Vento, "A comparison of algorithms for maximum common subgraph on randomly connected graphs," in *Structural, Syntactic, and Statistical Pattern Recognition*, ser. Lecture Notes in Computer Science, T. Caelli, A. Amin, R. Duin, D. de Ridder, and M. Kamel, Eds. Springer Berlin / Heidelberg, 2002, vol. 2396, pp. 85–106.

[84] W. Fan, J. Li, S. Ma, H. Wang, and Y. Wu, "Graph homomorphism revisited for graph matching," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 1161–1172, Sept. 2010.

[85] H. Bunke, "On a relation between graph edit distance and maximum common subgraph," *Pattern Recogn. Lett.*, vol. 18, no. 9, pp. 689–694, Aug. 1997.

[86] M. Neuhaus, K. Riesen, and H. Bunke, "Fast suboptimal algorithms for the computation of graph edit distance," in *Proc. 2006 joint IAPR int. conf. Structural, Syntactic, and Statistical Pattern Recognition*, 2006, pp. 163–172.

[87] Y. Zhu, L. Qin, J. X. Yu, Y. Ke, and X. Lin, "High efficiency and quality: large graphs matching," in *Proc. 20th ACM int. conf. Inform. and knowledge manage. (CIKM)*, ser. CIKM '11, New York, NY, USA, 2011, pp. 1755–1764.

[88] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," *Learning Theory and Kernel Machines*, pp. 129–143, 2003.

[89] N. Shervashidze, S. Vishwanathan, T. H. Petri, K. Mehlhorn, and K. M. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Proc. Intl. Workshop on Artificial Intelligence and Statistics Society for Artificial Intelligence and Statistics*, vol. 5, 2008, pp. 488–495.

[90] K. Riesen, X. Jiang, and H. Bunke, *Managing and Mining Graph Data*. Springer, 2010, ch. 5: Exact and Inexact Graph Matching.

[91] B. P. Kelley, R. Sharan, R. M. Karp, T. Sittler, D. E. Root, B. R. Stockwell, and T. Ideker, "Conserved pathways within bacteria and yeast as revealed by global protein network alignment," *Proc. Nat. Academy of Sci. (PNAS)*, vol. 100, no. 20, pp. 11 394–11 399, 2003.

[92] R. Sharan, S. Suthram, R. M. Kelley, T. Kuhn, S. McCuine, P. Uetz, T. Sittler, R. M. Karp, and T. Ideker, "Conserved patterns of protein interaction in multiple species," *Proc. Nat. Academy of Sci. (PNAS)*, vol. 102, no. 6, pp. 1974–1979, 2005.

[93] J. Flannick, A. Novak, B. S. Srinivasan, H. H. McAdams, and S. Batzoglou, "Graemlin: general and robust alignment of multiple large interaction networks," *Genome Research*, vol. 16, no. 9, pp. 1169–1181, 2006.

[94] M. Zaslavskiy, F. Bach, and J.-P. Vert, "A path following algorithm for the graph matching problem," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, pp. 2227–2242, 2009.

[95] M. Zaslavskiy, F. Bach, and J. Vert, "Global alignment of protein-protein interaction networks by graph matching methods," *Bioinformatics*, vol. 25, no. 12, pp. i259–i267, 2009.

[96] G. W. Klau, "A new graph-based method for pairwise global network alignment," *BMC bioinformatics*, vol. 10, no. Suppl 1, p. S59, 2009.

[97] R. Singh, J. Xu, and B. Berger, "Global alignment of multiple protein interaction networks with application to functional orthology detection," *Proc. Nat. Academy of Sci. (PNAS)*, vol. 105, no. 35, pp. 12 763–12 768, 2008.

[98] O. Kuchaiev, T. Milenković, V. Memišević, W. Hayes, and N. Pržulj, "Topological network alignment uncovers biological function and phylogeny," *J. Roy. Soc. Interface*, vol. 7, no. 50, pp. 1341–1354, 2010.

[99] O. Kuchaiev and N. Pržulj, "Integrative network alignment reveals large regions of global network similarity in yeast and human," *Bioinformatics*, vol. 27, no. 10, pp. 1390–1396, 2011.

[100] R. Preis, "Linear time 1/2 -approximation algorithm for maximum weighted matching in general graphs," in *Proc. 16th conf. Theoretical aspects of comput. sci. (STACS)*, ser. STACS'99. Berlin, Heidelberg: Springer-Verlag, 1999, pp. 259–269.

[101] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *IEEE Trans. Pattern Analysis and Machine Intelligence,*, vol. 10, no. 5, pp. 695–703, 1988.

[102] M. Ley, M. Herbstritt, M. R. Ackermann, O. Hoffmann, M. Wagner, S. von Keutz, K. Hostert, and D. Holzträger, *The DBLP Computer Science Bibliography*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Accessed April 2012. [Online]. Available: http://www.informatik.uni-trier.de/~ley/db/