A computationally driven comparative survey of network alignment, graph matching, and network comparison in pattern recognition and systems biology.

Marissa Graham

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Emily Evans, Chair
Benjamin Webb
Christopher Grant

Department of Mathematics

Brigham Young University

ABSTRACT

A computationally driven comparative survey of network alignment, graph matching, and network comparison in pattern recognition and systems biology.

Marissa Graham
Department of Mathematics, BYU
Master of Science

Comparative graph and network analysis plays an important role in both systems biology and pattern recognition, but existing surveys on the topic have historically ignored or underserved one or the other of these fields. We present a integrative introduction to the key objectives and methods of graph and network comparison in each, with the intent of remaining accessible to relative novices in order to mitigate the barrier to interdisciplinary idea crossover.

To guide our investigation, and to quantitatively justify our assertions about what the key objectives and methods of each field are, we have constructed a citation network containing 5,793 vertices from the full reference lists of over two hundred relevant papers, which we collected by searching Google Scholar for ten different network comparison-related search terms. This dataset is freely available on Github. We have investigated its basic statistics and community structure, and framed our presentation around the papers found to have high importance according to five different standard centrality measures. We have also made the code framework used to create and analyze our dataset available as documented Python classes and template Mathematica notebooks, so it can be used for a similarly computationally-driven investigation of any field.

Keywords: graph matching, graph alignment, graph comparison, graph similarity, graph isomorphism, network matching, network alignment, network comparison, network similarity, network alignment, comparative analysis, local alignment, global alignment, protein network, computational biology, biological networks, protein-protein interactions, computational graph theory, pattern recognition, exact graph matching, inexact graph matching, graph edit distance, graphlets, network motifs, graph matching algorithms, bipartite graph matching, node similarity, graph similarity search, attributed relational graphs

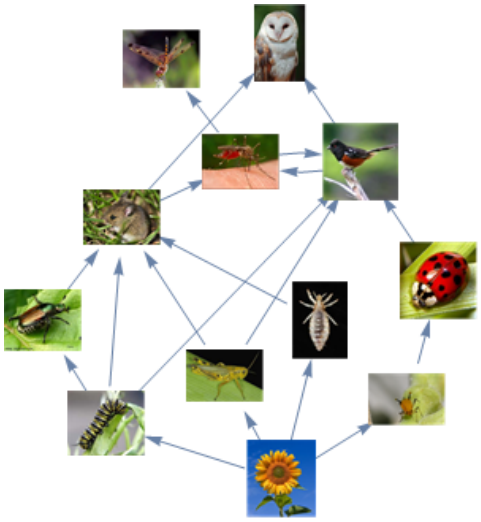
Contents

List of Tables

List of Figures

Chapter 1. Introduction

1.1 Motivation



"SimpleFoodWeb" Mathematica sample network.

Figure 1.1: Using a network to represent relationships in a food web.

Networks[1] are first and foremost a way to model the relationships between objects, which we do by representing objects as vertices and and relationships as edges. For example, we might use a graph to represent the relationships in a food web, or between characters in a successful movie franchise.

In some cases, using this representation simply to visualize relationships is useful, but we generally would also like to computationally exploit it in order to gain further insight about the system we are modeling.

As a first example, consider what questions we might ask about an infrastructure network such as a road network, phone lines, power grid, or the routers and fiber optic connections of the Internet itself. How do we efficiently get from here to there? How much traffic can flow through the network? What happens if an intersection is clogged, or a power plant fails?

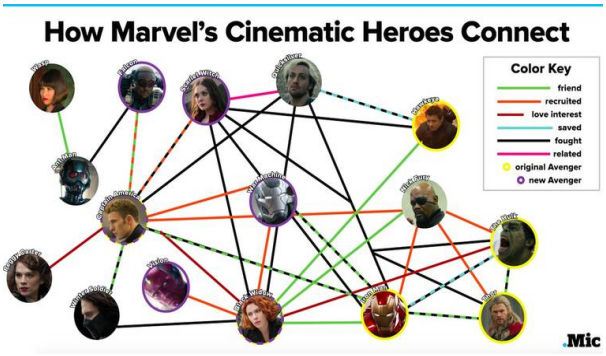


Diagram from the mic.com article "Here's How the Marvel Cinematic Universe's Heroes Connect–in One Surprising Map".

Figure 1.2: Using a network to represent relationships between movie characters.

We can also ask questions about social networks representing relationships between people. Who is the most important? Who controls the flow of information? To what extent do

[1]The term *network* is sometimes used interchangeably with the term *graph*. While they both refer to the same mathematical object, we attempt to follow the heuristic throughout of using the term *graph* to refer to a purely mathematical object and *network* to refer to a real-world system.

the people you consider your friends consider themselves *your* friend? How similar are you to your friends? To what extent are your friends friends with each other? How well is everybody connected to each other? To what extent do people form relationships with people who are like them? What do communities and strong friend groups look like, mathematically?

One common question across all of mathematics is how similar objects are to each other. With networks, we can ask this question about individual vertices in a network, but we also frequently want to ask it about networks themselves. For example, we might ask "How similar are you to other students, based on your friendships at school?", but we can also ask "Which proteins, protein interactions and groups of interactions are likely to have equivalent functions across species?" [72]

For objects as combinatorially complex as networks, similarity calculation is a difficult problem, the study of which has its roots in the 60s and 70s [17] and, as illustrated in Figure 1.3, has gained significant attention in the past twenty years as interesting network data becomes more readily available and computationally feasible. the problem has become more computationally feasible.

The goal of this project is to provide a broad outline of the study of network similarity, but without the help of prior expertise in the field, it is laborious at best and impossible at worst to know which works are important and which are irrelevant. Instead, we use the tools of network theory to study the network of citations between scientific papers on ~~a certa~~this in topic. This allows us to use standard network analysis techniques to determine which papers are the most important or influential, and has the additional advantage of bringing transparency to the process. That is, we can quantitatively justify our assertions using standard centrality and community detection measures, rather than relying on existing expertise in the field to give weight to our claims.

The remainder of this work is structured as follows: In the remainder of this chapter, we introduce the necessary mathematical background to inform our analysis of our citation network dataset. In Chapters 2 and 3, we introduce our citation network dataset. We analyze its basic structure, find two main fields of application within the study of network

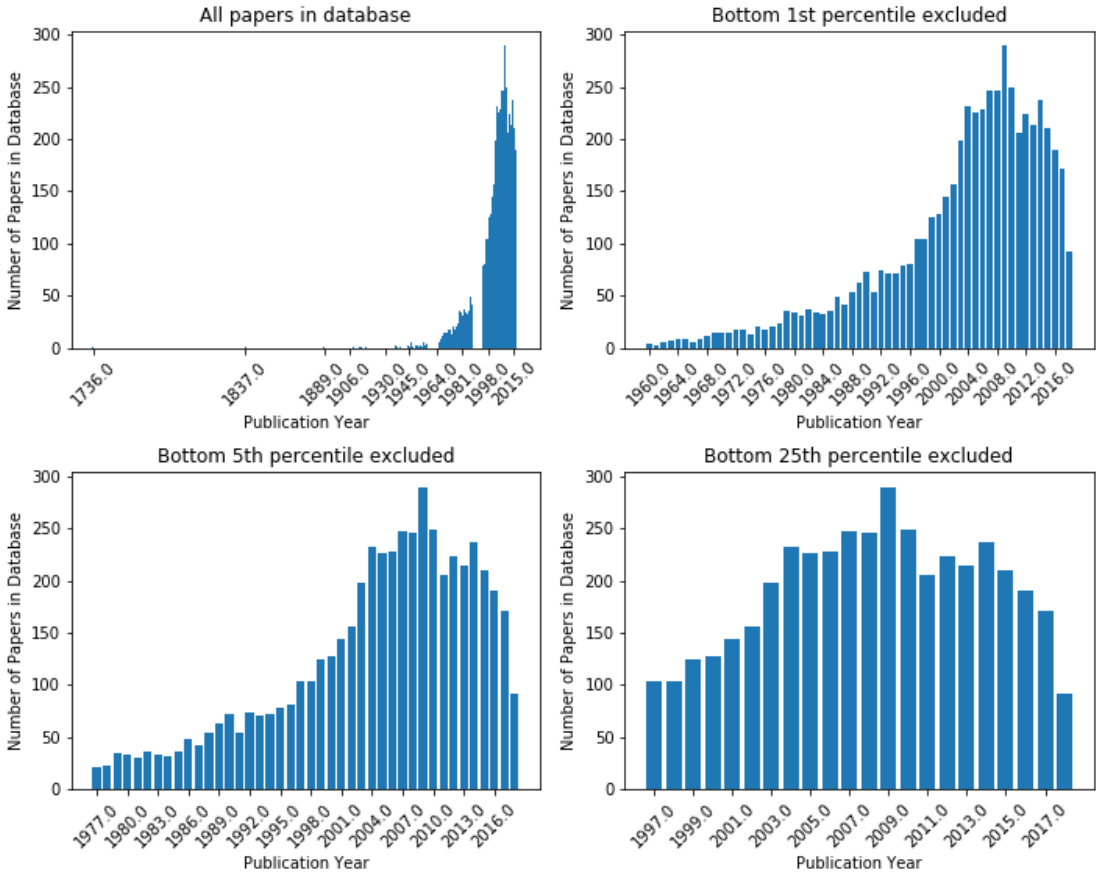


Figure 1.3: Distribution of papers published by year in our citation network of network similarity-related papers. Interestingly, year cutoffs for significant percentiles seem to roughly correspond to the spread of computers, personal computers, and the Internet, respectively.

comparison, and choose a reading list from the high centrality vertices in our dataset. In Chapters 4 and 5, we discuss our findings from this reading, and then conclude by discussing potential cross-applications between our two observed fields in Chapter 6.

1.2 Background

In this section we introduce the definitions and notation required to give context to our analysis of the citation network. Our presentation follows Newman's *Networks: An Introduction* [56] closely, with the remainder of definitions not otherwise cited sourced from *Algorithms and Models for Network Data and Link Analysis* [30]

A **graph** $G(V, E)$ is formally defined as a finite, nonempty set V of **nodes** or **vertices**,

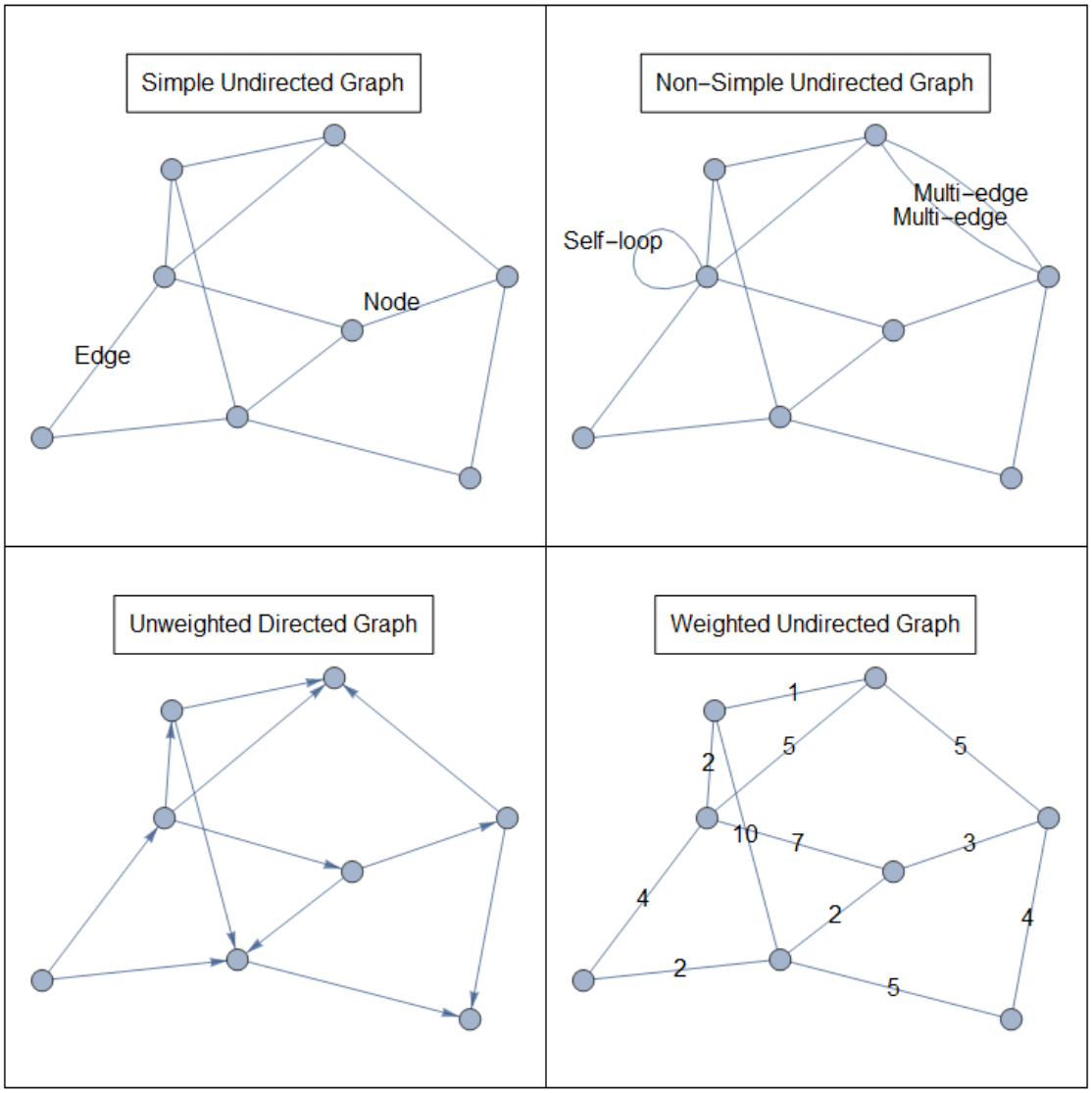


Figure 1.4: Basic types of networks

combined with a set $E \subset V \times V$ of **edges** representing relationships between pairs of vertices. Throughout this work, we denote the number of vertices in a graph by n and the number of edges by m where not otherwise specified.

In this work we will deal with **simple graphs**, which are those that do not have more than one edge between any pair of vertices (that is, a **multiedge**[2]), and do not have any edges from a vertex to itself (a **self-edge** or **self-loop**).

We also are concerned with whether a graph is **directed** or **undirected**. In an undirected

[2]A multigraph without edge loops is not simple, but if we represented the number of edges between pairs of vertices as edge weights instead of multiedges, that would be a simple graph.

graph, we have an edge *between* two vertices, whereas in a directed graph we have edges *from* one vertex *to* another vertex. Throughout this work, we will use the notation $v_i \leftrightarrow v_j$ for an undirected edge between vertices v_i and v_j, and $v_i \rightarrow v_j$ for a directed edge. In either case, the edge $v_i \leftrightarrow v_j$ or $v_i \rightarrow v_j$ is **incident** to vertices v_i and v_j, and vertices v_i and v_j are therefore considered **neighbors**.

A graph can also be **weighted**, meaning each edge is assigned some real, generally positive value w_{ij} representing the "strength" of the connection between vertices v_i and v_j.

In an undirected graph, the **degree** of a vertex is the sum of the weights of the incident edges, and in a directed graph, the **indegree** and **outdegree** are the total weight of a vertex's incoming and outgoing edges, respectively. If the graph is unweighted, this is simply the number of adjacent, incoming, or outgoing edges, as the weight of each edge is one.

When studying real-world networks, we also make a distinction between **deterministic** and **random** networks. This distinction is roughly the same as that between a variable and a random variable. The vertices and edges in a deterministic network are "fixed", while in a random network, they need to be inferred from data using statistical inference methods. For example, our citation network is deterministic, but a network of protein interactions for a given species is not, as it must be inferred from experimental data on a limited number of members of that species.

1.2.1 Computational network properties.

Whether a network is directed or weighted or simple will inform our approach to its analysis, but these properties are generally included as metadata rather than computationally determined. The remainder of the properties we consider in network analysis are determined computationally, with varying degrees of algorithmic complexity.

Connectivity. One example of interesting computational network properties is the question of whether a network is **connected**, as illustrated in Figure 1.6; that is, there is a **path** between any pair of vertices, where a path is defined to be a sequence of vertices such that consecutive vertices are connected by an edge. The **length** of a path is the number of edges

connecting the vertices in the sequence. In the case of a directed graph, we make the distinction between weak and strong connectivity. A **weakly connected graph** is one which is connected when each edge is considered as undirected, while a **strongly connected graph** requires a path from every vertex to every other vertex, even while respecting edge directions.

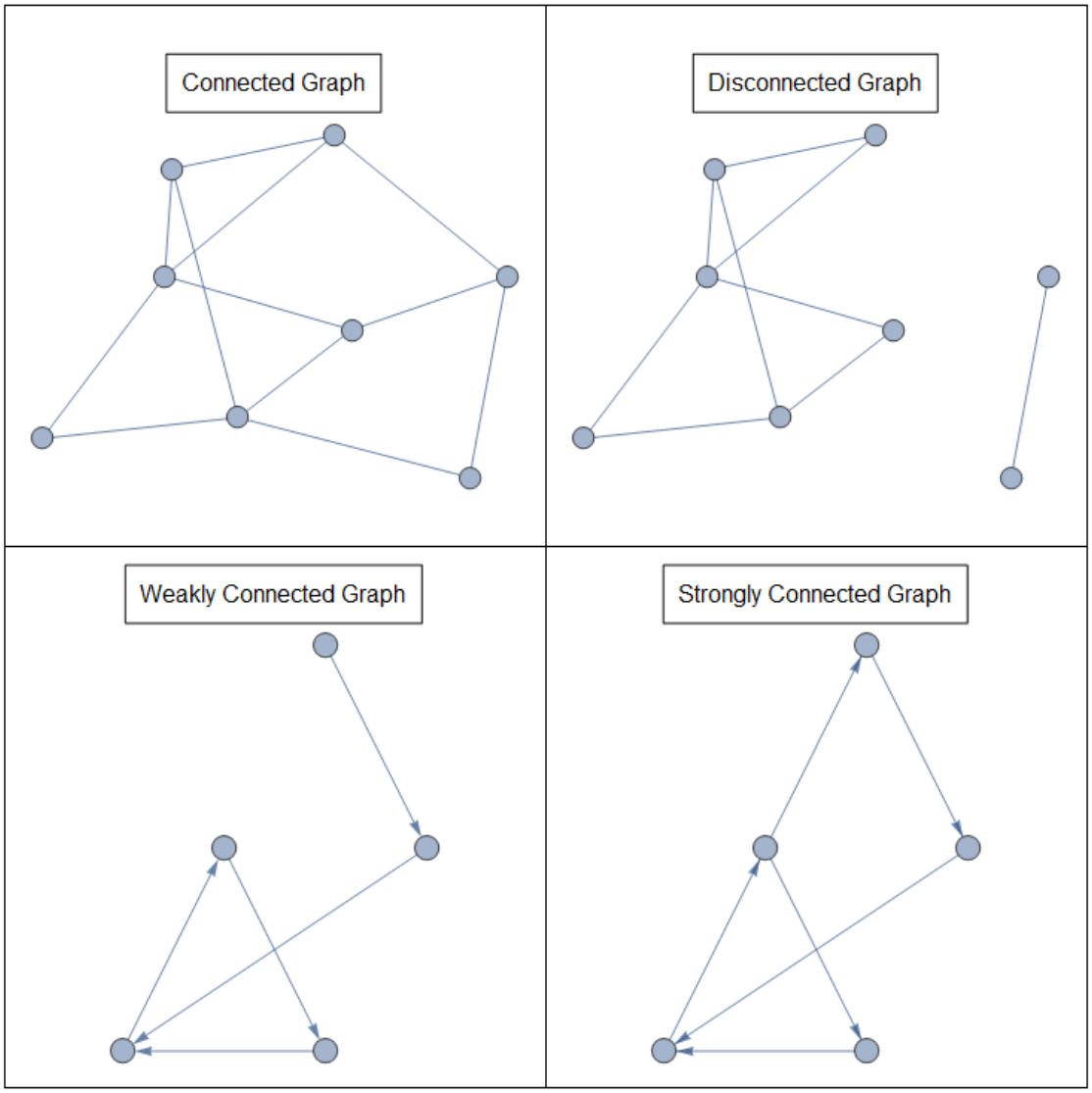


Figure 1.6: Simple examples of different types of network connectivity.

If an undirected network is not connected, or a directed network, is not weakly connected, the network has multiple **connected components** or **weakly connected components**. Each component is a subset of vertices such that there is a path between every pair of member vertices, and no paths between any member and a nonmember. For example, the

disconnected graph in Figure 1.6 has two components. The weakly connected components of a directed graph are the components in the corresponding undirected network.

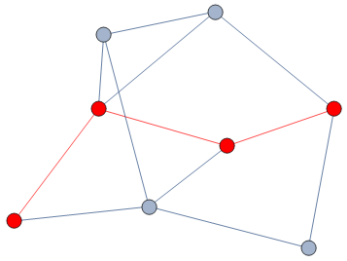
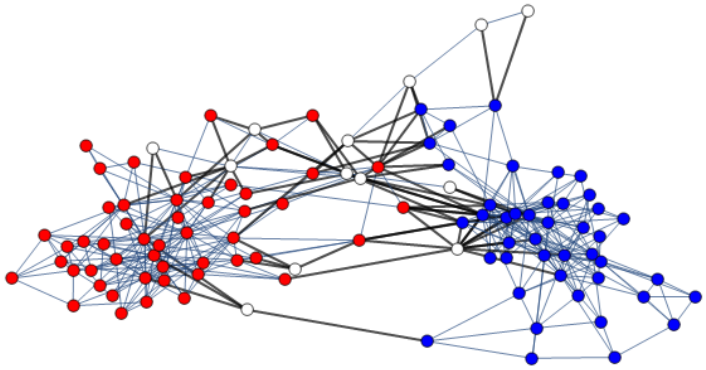


Figure 1.5: A path of length three on a small network.

In a typical real world network, there is generally a single large component or weakly connected component which contains most of the vertices, with the rest of the vertices contained in many small disconnected components. We refer to this large component as the **giant component**, and its relative size gives us a measure for how "close" a network is to being connected; the higher the percentage of vertices are in the giant component, the closer the network is to being connected.



"USPoliticsBooks" Mathematica sample network.

Figure 1.7: A network of U.S. politics books. Vertices categorized as liberal, conservative, or neutral are colored blue, red, and white, respectively, and edges that run between different categories are bolded.

Assortativity. We can also consider whether a network is **assortative**. That is, if the vertices in the network have some discrete-valued property, we ask whether the edges in the network are more likely to run between vertices of the same type. If all of the edges run between vertices of the same type, the assortativity of the network is 1; if all edges run between edges of different types, the assortativity is -1.

For example, the network of U.S. politics books in Figure 1.7 is strongly assortative with an assortativity value of 0.72. Most of the connections are between books with the same political classification, which we can visually confirm by coloring the vertices accordingly and highlighting the few edges of the graph that run between books with different classifications.

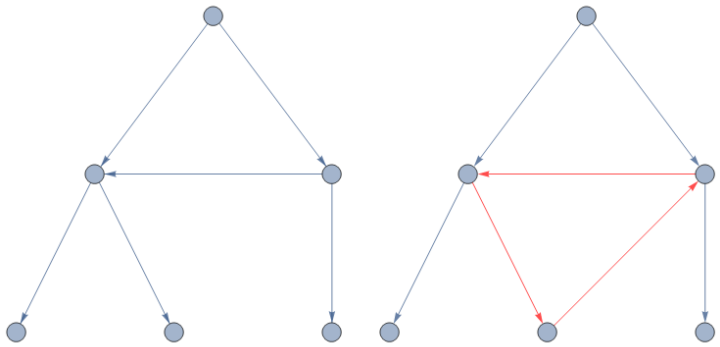


Figure 1.8: An acyclic directed network (left) vs. one which contains a cycle (right).

Acyclic networks. We also consider whether a directed network is **acyclic**, meaning that it contains no **cycles** or nontrivial paths from any vertex to itself.

Our most important example of a directed acyclic network is a **citation network**. In a citation network, we include an edge from a paper to each reference it cites. Any cycle in this network would require edges both from a newer paper to an older paper, and from an older paper to a newer paper. If we only cite papers which have already been written, which is the case for the papers in our dataset and generally true for academia as a whole, we cannot have any cycles.

Chapter 2. Creating the citation network

2.1 Approach

Citation network creation is not a trivial task. Although some journals and databases provide a citation network of the references in their own domain, since our approach considers a highly interdisciplinary field, this approach discards large sections of our desired network. As a result of this, and since intellectual property restrictions preclude simply scraping an entire citation network, we constructed the dataset manually by collecting reference lists for relevant papers and then building the network accordingly.

Relevant papers were found by searching google scholar for "graph" or "network" +

"alignment", "comparison", "similarity", "isomorphism", or "matching"[1] Topic-relevant papers were initially collected from the first five pages of results for each of the ten search terms on May 4th, 2018, after which new papers published through June 25th, 2018 were collected from a google scholar email alert for those same ten search terms. For each of these papers, we stored the plaintext reference list in a standardized format which could be easily split into the individual freeform citations. Any paper for which we have a reference list is referred to as a "parent" record, and the references are referred to as the "child" records. In total, we collected 7,790 child references from 221 parent papers.

In order to create the network, we needed to parse the freeform citations for each reference list to obtain metadata and recognize records as repeatedly cited. This is a difficult problem, as the records in the database span several hundred years and represent a wide variety of citation styles and languages, as well as significant optical character recognition and Unicode-related challenges. Instead of attempting to parse a citation into component parts, we used the REST API to search for each record in the CrossRef database, which already has the metadata parsed for any record it includes. We marked results as duplicate if their metadata matches and both are known to be correct, or if both their metadata and original freeform citation match exactly.

The results given by the CrossRef API are considered correct if the title of the record can be found in the original freeform citation, and unverified otherwise. We were able to automatically verify results for about 75% of the parent records, and about half of their children. We are conservative about marking records as duplicate, which means having so many unverified records dramatically misrepresents the structure of the network. We therefore went through the approximately three thousand unverified records by hand.

For unverified parents, we manually corrected or found title, year, author, DOI number if existent, and URL information as well as reference and citation counts. For unverified child references, we first went through and marked any correct but unverified results. We found about half of the unverified child results to be correct, despite being unable to be

[1]Future work should probably include "graph kernel(s)" in this list.

automatically verified due to punctuation discrepancies, misspellings, unicode issues, or citation styles that do not include the title. Next, results were counted as correct (but noted as "half-right") if the CrossRef API returned a review, purchase listing or similar for the correct record. For the remaining incorrect references, we manually parsed the author, title, and year from the citation, or looked them up if not included. Finally, we deleted any records which did not refer to a written work of some kind; specifically, references simply citing a website, web service, database, software package/library, programming language, or "personal communication".

We then wrote the entire citation network to a GML file which can be loaded in Mathematica. By default, the code used to generate the GML file includes the title, year, reference and citation counts for each record as vertex properties. Including further metadata as vertex properties is not difficult, but additional string-valued properties dramatically slow Mathematica's ability to load such a large network[2], and a network as large as ours cannot be loaded at all, and so we do not include any more of them than strictly necessary.

The dataset itself and the code and source files used to generate it can be found on GITHUB REPOSITORY LINK, as well as documentation and instructions for using it to generate a similar dataset for any collection of properly-formatted reference list files.

2.2 Basic statistics

Construction-related issues. Our full citation network contains a total of 7,491 references between 5,793 papers. This results in a fairly low mean degree, or average number of references per paper, which is due to an inherent limitation in the construction of almost any citation network. We can include all the references for a small group of papers, but including all the references for *their* references and so on is an exponentially more expensive task, and we therefore generally only include children for a small fraction of the total vertices.

[2]The network contains 5,793 vertices and 7,491 edges, and takes almost exactly two minutes to load on a 2.6GHz 6th-gen quad core Intel Core i7 CPU with 16GB of RAM running Windows 10 using Mathematica 11.2.

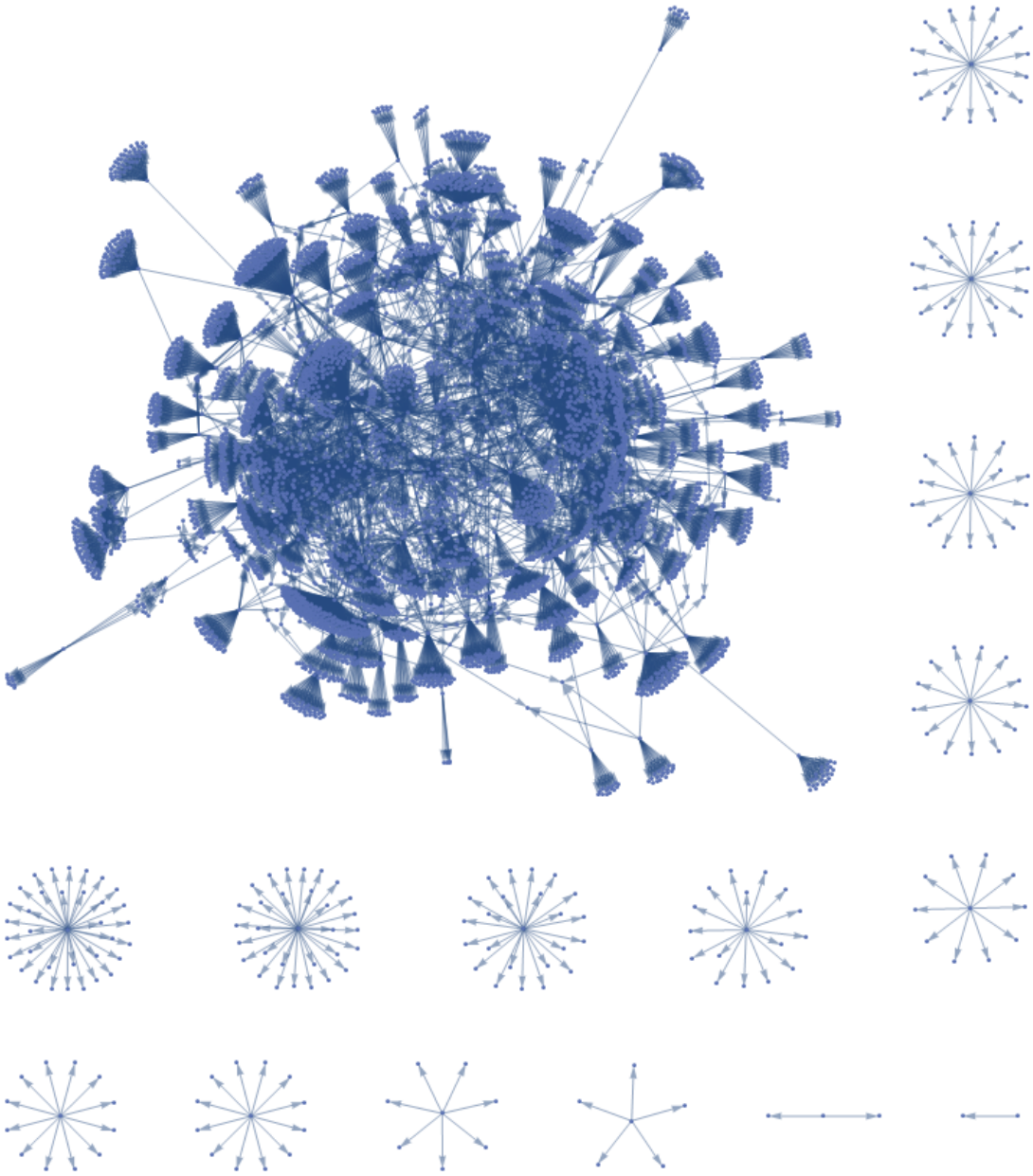


Figure 2.1: The full citation network of the dataset used for the project.

Since the edges in our network are hand-constructed using individual reference lists, it includes an abnormally small fraction of vertices with children. A typical citation network, such as the SciMet and Zewail datasets which are displayed in Appendix Figures A.1 and A.2, respectively, and whose analysis is included in Table 2.1, is constructed by scraping a single database. This results in a much higher fraction of children whose references are included, but any references which are not in the database in question are missed, so the mean degree is still quite low.

	G	G_p	sciMet	zewail	R	R_d
Vertices	5793	1062	1092	3145	5793	5793
Edges	7491	2775	1308	3743	7491	7491
Mean degree	1.29	2.61	1.20	1.19	1.29	1.29
Fraction with children	0.038	0.193	0.523	0.599	0.733	0.038
Diameter	10	9	14	22	21	9
Connected components	16	1	114	281	504	3
Fraction in giant component	0.960	1.000	0.784	0.797	0.900	0.999

Table 2.1: Comparing statistics for our dataset to other networks. The Mathematica code used to load and construct all six networks can be found in Table **??**.

The pruned network G_p. We also consider the *pruned network*, shown in Figure 2.2, which is defined to be the giant component of the subnetwork of vertices with positive outdegree or indegree greater than one. We do so because the main purpose of our dataset is to determine which papers are important in the field of network similarity, but the vast majority of references in the database are only cited by one paper and frequently have very little relevance to network similarity itself.

To reduce the influence of off-topic papers on our results, we restrict our network to ~~both~~ our parent vertices, which we have hand-curated to be relevant to network similarity, and all vertices which are cited by more than one parent paper. This shrinks the number of vertices by a factor of almost six, correspondingly raises the fraction of vertices with children, and approximately doubles the mean degree.

Figure 2.2: The pruned citation network.

Comparison to other networks. In Table 2.1, we calculate the mean degree, fraction of vertices with children, diameter, number of connected components, and fraction of vertices in the giant component for six different networks: our full network G, its pruned version G_p, two datasets from the Garfield citation network collection, a uniformly generated directed random graph R, and a random graph R_d with the same degree sequences as G. The random network R is generated from a uniform distribution. The other, R_d, is constructed to match the degree sequence of G, by assigning each vertex "stubs" according to the desired number of incoming and outgoing edges and then matching them by uniformly sampling the available stubs.

Connectivity. Our full network displays a high level of connectivity; 96% of vertices are contained in the giant component, and it has only 16 connected components, compared to 90% containment in the giant component and 504 connected components for a randomly generated network of the same size. The diameter is also low compared to a random network and to our choices of real-world network. Since our network consists of papers collected on a specific topic, which have an outside reason to cite the same papers, this high level of connectivity is not surprising.

The construction of the network also explains the high connectivity compared to the real-world datasets, which only have about 80% of their vertices in their giant components; if the generation of the citation network is limited to a single database, as the sciMet and zewail datasets are, cocitation connections in other databases will be lost. This makes it more difficult for the giant component to fill the network, and results in longer paths between connected vertices.

The only dataset tested with better connectivity than ours is the random network R_d, which has almost complete containment–99.9%–in the giant component. This is not surprising. Approximately speaking, in order for a parent vertex to be disconnected from the giant component, its children must all have exactly one parent, and no children. In a real-world network, this is easier to find, since a paper's references are not randomly selected. A single work from an author or topic which is relatively disconnected to the rest of the network similarity academic community can generate a significant number of references which are not in the giant component. By contrast, since about 82% of the vertices in G have exactly one parent and no children, the probability of a parent vertex with outdegree n being disconnected from the giant component is $(0.82)^n$, or $(0.82)^{26.2} \approx 0.5\%$ for the mean outdegree of the parents. This probability shrinks further as the number of disconnected parent vertices grows, as fewer single-parent vertices become available compared to the rest.

2.3 Centrality

The main goal in creating the citation network is to determine which papers are most important or **central**. The question of which vertices are the most central to a network is widely researched in network theory, and there correspondingly exists a wide variety of centrality measures used to quantify different ideas about importance. For this project, we chose five centrality measures that we expect to coincide with an intuitive definition of which papers in the network are the most relevant.

Indegree. This measures the number of times each paper was cited by the parent papers in our network. Since the parent papers approximately represent everything determined to be most relevant by google scholar in a search for network comparison-related search terms[3], the top values for indegree should give us a rough idea of which papers are formative for the field and therefore more frequently cited by the parent papers.

Outdegree. Since the pruned network consists of the parent papers as well as the papers that appear in more than one parent's reference list, this measures the number of a parent's references which have been cited by other parents in the network. The top values for outdegree should therefore give us an idea of which papers survey the most well-known topics in the field.

Betweenness. Betweenness centrality measures the extent to which a vertex lies on paths between other vertices (sentence comes straight from newman). Intuitively, this should correspond to papers which make uncommon connections between other works; either those that are applicable to a wide variety of fields and applications, or those which build on disparate ideas in an original way. Unsurprisingly, we see some overlap between the papers with the highest outdegree and those with the highest betweenness, since the goal of a survey is to discuss a wider variety of ideas than an original paper (which only cites the specific works it builds on) can reasonably include.

[3]This is somewhat skewed by our inclusion of newly published papers collected from an email alert after the initial search, which may not be the most relevant overall.

Closeness. Recall that a path between two vertices is a sequence of vertices such that consecutive vertices are connected by an edge; a **geodesic path** is the shortest possible path between any two vertices, and its length is the geodesic distance. We define **closeness** to be the inverse of the average geodesic distance between a vertex and all other vertices in the network. Closeness centrality therefore takes on the highest values for a vertex which is a short average distance from other vertices. For example, in Figure 2.3 the highlighted vertex on the left has closeness centrality 1, since it has a distance of 1 from all other vertices. On the right, the average geodesic distance from the highlighted vertex to the other six is $\frac{1}{6}(1+1+2+2+2+2) = \frac{5}{3}$, so the closeness centrality is 0.6.

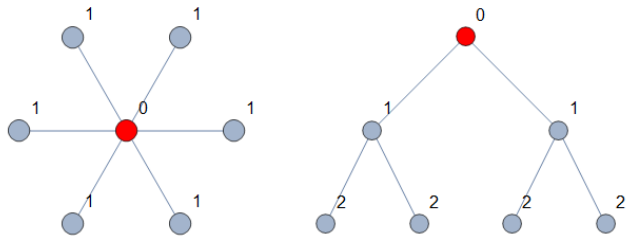


Figure 2.3: Two graphs with vertices labeled by their geodesic distance from the highlighted vertex.

In a friendship network, this corresponds to a person who seems to know just about everybody. Similarly, a paper in our citation network will have higher closeness centrality if it only takes a few steps through a paper's reference list or citations to another paper's reference list or citations to get to every other paper in the network.

HITS. For a directed network such as the citation networks used here, we may want to separately consider the notion that a vertex is important if it points to other important vertices, and the notion that a vertex is important if it is pointed to by other important vertices. This is the goal of the HITS or hyperlink-induced topic search algorithm. We define two different types of centrality for each vertex. We have **authority centrality**, which measures whether a specific vertex is being pointed to by vertices with high **hub centrality**, which in turn measures whether a specific vertex points to vertices with high authority centrality. By defining the hub and authority centralities of a vertex to be proportional to the sum of the authority and hub centralities, respectively, of its neighbors, this definition reduces to a pair of eigenvalue equations which can be easily solved numerically.

That is, if x_i is the authority centrality of the i-th vertex in a network, y_i is the hub centrality of the j-th vertex, A_{ij} is the weight of the edge from j to i if it exists, and 0 otherwise, and α, β are proportionality constants, we have

$$x_i = \alpha \sum_j A_{ij} y_j \text{ and } y_i = \beta \sum_j A_{ji} x_j.$$

2.3.1 High centrality vertices. We can observe that the pruned network, shown in Figure 2.2, seems to contain two clusters of more tightly connected vertices.[4] We would like to collect the important papers for both the network as a whole and for the distinct communities it contains, so we partition the dataset in half using a modularity maximizing partition; that is, we choose two groups of vertices such that the fraction of edges running between vertices in different groups is minimized.

For both the pruned network and the two halves of our partition, we collect the top ten papers according to these five different centrality measures and summarize the results in the following tables. Since the numerical values for indegree and outdegree have intuitive meaning, we report the value itself. However, the values for betweenness, closeness, and the two HITS centralities are unintuitive, context-free real numbers, so we report the rank of each paper with respect to each measure rather than the actual value. We also calculate betweenness and closeness for the undirected version of the network, to allow those rankings to be based on citing relationships in either direction.

The papers in each table are sorted from maximum to minimum according to

$$f(p) = \frac{k_p^{in}}{k_{max}^{in}} + \frac{k_p^{out}}{k_{max}^{out}} + \sum_{i=1}^{4} \frac{1}{r_i(p)},$$

where k_p^{in} and k_p^{out} are the indegree and outdegree of a paper p, the maximums of which are taken with respect to the pruned network or partition half in question, and $r_i(p)$ is the rank of a paper p according to the i-th of our four centrality metrics, which is defined to be

[4] We discuss the motivation behind and significance of this partition in detail in Chapter 3.

infinity if a paper is not in the top ten for that metric.

Title	Indegree	Outdegree	Betweenness	Closeness	HITS Auth.	HITS Hub
◊Thirty Years of Graph Matching in Pattern Recognition [17]	20*	109*	1	2		1
†Fifty years of graph matching, network alignment and network comparison [26]	6	71*	2	1		3
†Networks for systems biology: conceptual connection of data and function [25]	2	102*	3	3		2
◊An Algorithm for Subgraph Isomorphism [76]	20*	4	7	4	1	
†Modeling cellular machinery through biological network comparison [72]	9	41*	8			
◊Computers and Intractability: A Guide to the Theory of NP-Completeness [35]	16*	0	4	5		
◊The graph matching problem [50]	2	55*	5	6		7
†A new graph-based method for pairwise global network alignment [41]	9	13		8		
†On Graph Kernels: Hardness Results and Efficient Alternatives [33]	11	10	6			
◊Error correcting graph matching: on the influence of the underlying cost function [9]	10	16		7	7	8
◊A graduated assignment algorithm for graph matching [31]	18*	0			5	
◊The Hungarian method for the assignment problem [46]	17*	0				
◊An eigendecomposition approach to weighted graph matching problems [77]	15*	5			6	
◊Recent developments in graph matching [7]	1	51*				4
†MAGNA: Maximizing Accuracy in Global Network Alignment [68]	5	35*				
◊A distance measure between attributed relational graphs for pattern recognition [67]	14*	0			3	
†Pairwise Global Alignment of Protein Interaction Networks by Matching Neighborhood Topology [75]	13*	0				
†Topological network alignment uncovers biological function and phylogeny [11]	12*	0				
A graph distance metric based on the maximal common subgraph [45]	10	0		10	4	
◊Efficient Graph Matching Algorithms [52]	0	43*				5
Local graph alignment and motif search in biological networks [5]	8	10	10			
†Global alignment of multiple protein interaction networks with application to functional orthology detection [74]	11*	0				
On a relation between graph edit distance and maximum common subgraph [8]	11	0			2	
◊Graph matching applications in pattern recognition and image processing [16]	0	40*				6
◊Fast and Scalable Approximate Spectral Matching for Higher Order Graph Matching [58]	0	41*	9			
◊Structural matching in computer vision using probabilistic relaxation [14]	9	0			10	
◊A new algorithm for subgraph optimal isomorphism [24]	2	21				9
BIG-ALIGN: Fast Bipartite Graph Alignment [43]	2	21		9		
◊A graph distance measure for image analysis [27]	8	0			8	
A new algorithm for error-tolerant subgraph isomorphism detection [53]	8	0			9	
◊A (sub)graph isomorphism algorithm for matching large graphs [18]	3	16				10

†Also top for Group 1 (biology dominated); ◊Also top for Group 2 (computer science dominated); *Top ten for indegree/outdegree

Table 2.2: Highest centrality papers for the entire pruned network.

Title	Indegree	Outdegree	Betweenness	Closeness	HITS Auth.	HITS Hub
◊Networks for systems biology: conceptual connection of data and function [25]	2	90*	1	2		1
◊Fifty years of graph matching, network alignment and network comparison [26]	4	56*	2	1		2
◊Modeling cellular machinery through biological network comparison [72]	9	40*	4	3	10	9
◊MAGNA: Maximizing Accuracy in Global Network Alignment [68]	5	35*	7	6		3
◊On Graph Kernels: Hardness Results and Efficient Alternatives [33]	10*	9	3	8		
Biological network comparison using graphlet degree distribution [62]	11*	0		7	4	7
◊A new graph-based method for pairwise global network alignment [41]	8	12	9	4	6	
Network Motifs: Simple Building Blocks of Complex Networks [54]	11*	0		9	8	
◊Pairwise Global Alignment of Protein Interaction Networks by Matching Neighborhood Topology [75]	12*	0			3	
◊Topological network alignment uncovers biological function and phylogeny [45]	12*	0			2	
NETAL: a new graph-based method for global alignment of protein-protein interaction networks [57]	6	26*				5
Collective dynamics of "small-world" networks [79]	10*	0		10	5	
Global network alignment using multiscale spectral signatures [59]	11*	0			9	
◊Global alignment of multiple protein interaction networks with application to functional orthology detection [74]	10*	0				
Conserved patterns of protein interaction in multiple species [71]	10*	0			7	
Pairwise Alignment of Protein Interaction Networks [44]	10*	0			1	
Alignment-free protein interaction network comparison [3]	2	22	6	5		
Graphlet-based measures are suitable for biological network comparison [36]	1	30*				8
Survey on the Graph Alignment Problem and a Benchmark of Suitable Algorithms [21]	0	26				4
Predicting Graph Categories from Structural Properties [12]	0	30*	5			
Fast parallel algorithms for graph similarity and matching [42]	1	23				6
Complex network measures of brain connectivity: Uses and interpretations [66]	0	28*	8			
Graph-based methods for analysing networks in cell biology [1]	0	30*				10
Demadroid: Object Reference Graph-Based Malware Detection in Android [78]	0	25	10			
Early Estimation Model for 3D-Discrete Indian Sign Language Recognition Using Graph Matching [48]	0	29*				
Indian sign language recognition using graph matching on 3D motion captured signs [47]	0	29*				

◊Also a top-centrality paper for the entire network; ; *Top ten for indegree/outdegree

Table 2.3: Highest centrality papers for Group 1 (biology dominated) in our partition of the pruned network.

Title	Indegree	Outdegree	Betweenness	Closeness	HITS Auth.	HITS Hub
◊Thirty Years of Graph Matching in Pattern Recognition [17]	17*	107*	1	1		1
◊An Algorithm for Subgraph Isomorphism [76]	15*	2	10	5	2	
◊A graduated assignment algorithm for graph matching [31]	18*	0	7	4	3	
◊An eigendecomposition approach to weighted graph matching problems [77]	15*	5		2	4	
◊The graph matching problem [50]	2	36*	3	3		8
◊A distance measure between attributed relational graphs for pattern recognition [67]	13*	0		7	1	
◊Recent developments in graph matching [7]	0	50*	8			2
◊Error correcting graph matching: on the influence of the underlying cost function [9]	9*	16		8		6
◊Fast and Scalable Approximate Spectral Matching for Higher Order Graph Matching [58]	0	41*	2			
◊Efficient Graph Matching Algorithms [52]	0	42*	5			4
◊Computers and Intractability: A Guide to the Theory of NP-Completeness [35]	11*	0	6			
◊The Hungarian method for the assignment problem [46]	14*	0				
◊Graph matching applications in pattern recognition and image processing [16]	0	40*				3
Efficient Graph Similarity Search Over Large Graph Databases [89]	0	28*	4	6		
A linear programming approach for the weighted graph matching problem [4]	8	8		9	9	
◊Structural matching in computer vision using probabilistic relaxation [14]	9*	0			5	
◊A graph distance measure for image analysis [27]	8	0			6	
Inexact graph matching for structural pattern recognition [10]	10*	0				
◊A new algorithm for subgraph optimal isomorphism [24]	2	21				5
Approximate graph edit distance computation by means of bipartite graph matching [64]	9	0				
Linear time algorithm for isomorphism of planar graphs (Preliminary Report) [37]	9	0				
Structural Descriptions and Inexact Matching [70]	9	0			7	
◊A (sub)graph isomorphism algorithm for matching large graphs [18]	3	16				7
A Probabilistic Approach to Spectral Graph Matching [22]	0	25*	9	10		
Hierarchical attributed graph representation and recognition of handwritten chinese characters [51]	6	0			8	
Exact and approximate graph matching using random walks [32]	1	14				9
A shape analysis model with applications to a character recognition system [65]	5	0			10	
Fast computation of Bipartite graph matching [69]	1	23*				
Graph Matching Based on Node Signatures [39]	0	17				10
Unsupervised Domain Adaptation Using Regularized Hyper-Graph Matching [19]	0	22*				

◊Also a top-centrality paper for the entire network; ; *Top ten for indegree/outdegree

Table 2.4: Highest centrality papers for Group 2 (computer science dominated) in our partition of the pruned network.

Chapter 3. Partitioning the citation network

3.1 The tagging and partitioning process

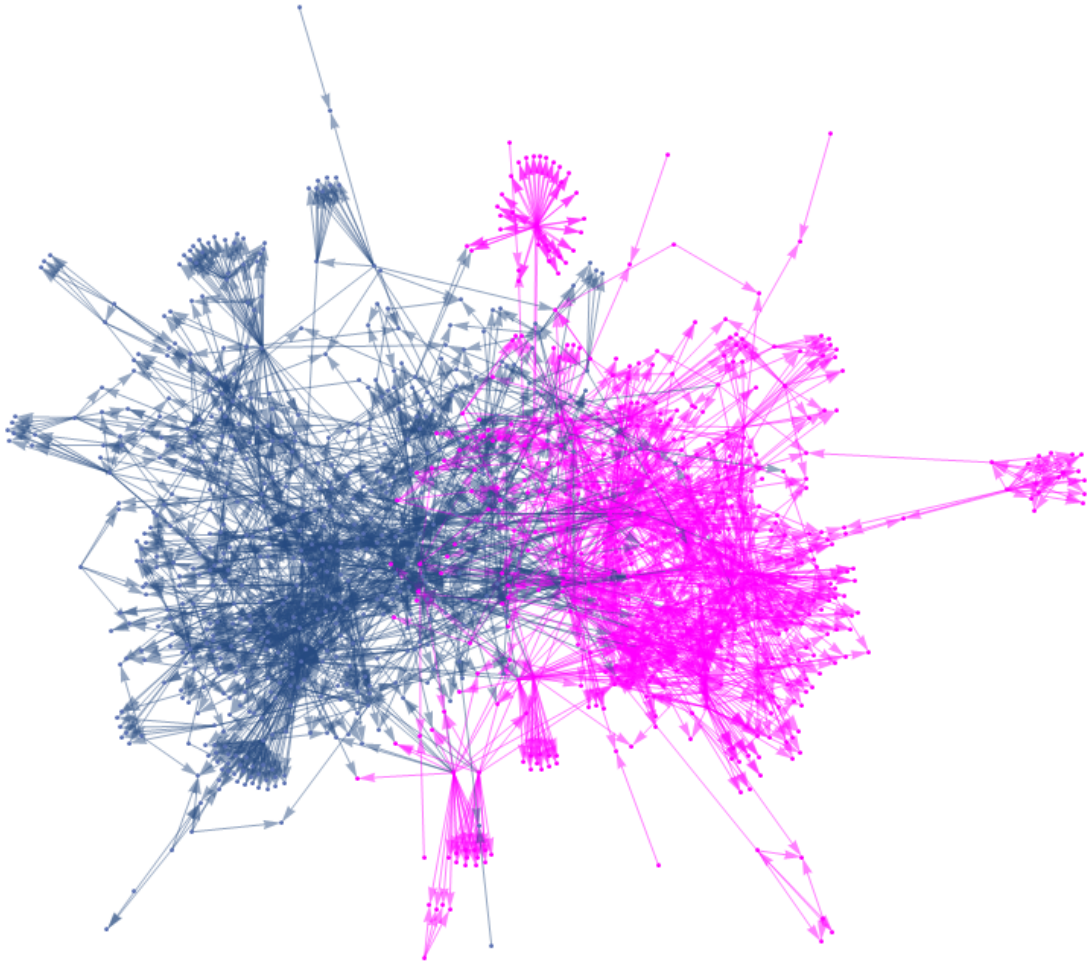


Figure 3.1: The two halves of our partition of the pruned network.

In the process of collecting relevant papers for our citation network, we noticed that network similarity applications seem to be almost exclusively found in the fields of biology and computer science[1], and we would therefore like to investigate the structure of the network

[1]While the "computer science" papers in our reading list all fall into the category of "pattern recognition", this is not necessarily the case for their references, and we therefore use the broader label in our subject tagging process. Similarly, we use "biology" instead of "systems biology" when describing our category labels.

I'm not sure this is strictly necessary.

with respect to these two categories.

Unfortunately, the metadata for the papers in our network does not include the subject information we would need to simply partition the network using on these categories; while the CrossRef API does sometimes include a "subject" category, it is present in less than 1% of items, and with almost six thousand references in our database, it is impractical to categorize their subjects by hand.

Instead, we partition the network into two categories of equal size. If we choose our partition in a way that minimizes the fraction of edges running between its two groups, it will preserve and separate the two clusters of more densely connected vertices first observed in Figure 2.2, as we can see in Figure 3.1. We then set out to determine whether these two halves of the network correspond to the two fields of study we noticed while constructing the dataset.

To do this, we need some way to roughly tag papers by their subject. We chose to do according to their journal(s) of publication, since this information is available for over 97% of the papers in our network[2]. There were a total of 2,285 unique journal names, which we tagged as "Computer Science", "Biology", and "Mathematics" according to the keywords listed in Table A.4. This strategy allowed us to quickly tag the majority of the papers as at least one of these three subjects.

Our journal-based tagging is a drastic improvement over the subject information provided by CrossRef, giving us information for about 67% of the total papers, and 53% of those in the pruned network. We found this to be sufficient to confirm our initial suspicions that the two communities observed in the network do in fact correspond to the fields of computer science (primarily pattern recognition) and systems biology. In the remainder this chapter, we show how these categories are reflected in the structure of the dataset, both overall and with respect to our partition, and then discuss the advantages that our dataset and analysis

[2]Journal information was not manually corrected for the papers for which the CrossRef API returned an incorrect result. However, in the vast majority of those cases, the result returned was very similar to the correct one–i.e., written by most of the same authors, or an older paper on the same subject. The subject information should therefore still be accurate enough for our purposes.

provide in our reading and writing process.

3.2 Results

Our first step is to color code the vertices in the pruned network according to their subject, as shown in Figure 3.2, so we can get a visual sense of how our tagged subjects are spread through the network. While the main two categories we are interested in are computer science and biology, we have also tagged the mathematics papers, so that we have a third category of similar size and generality as the other two. This serves as a control group, and allows us to consider and reject the hypothesis that there are three main subnetworks of similar papers instead of two.

Intuitively, we can see that the red and blue vertices are mostly clustered together on the two halves of the pruned network, confirming our suspicion that the two dense clusters of vertices we see in Figure 2.2 correspond to the two categories we observed while constructing the dataset. We also notice that the cluster of blue vertices only fills about half of its side of the partition, indicating that the biology category is significantly smaller than the computer science category. The yellow seems to be about evenly spread across the two halves, meaning that we do not have three distinct meaningful subnetworks of papers.

There also appear to be significantly more untagged vertices on the biology side of the partition. It is likely that this is not only because the computer science category is inherently larger, but because its papers are more likely to be tagged as such. A full half of the papers in the computer science category are published in an ACM, IEEE, or SIAM[3] journal (IEEE journals alone represent 35% of the CS-tagged papers), all of which are easily tagged using these acronyms as a keyword. There were not any analagous dominant organizations with acronym keywords for the biology journals in our dataset, so the tagging relies on topical keywords and therefore can identify fewer of the biology papers using a reasonable number of keywords in our search. As a result of this, the computer science network is more strongly

[3]Both "SIAM" and "algorithm" were used as keywords for both math and computer science, which accounts for about half of the overlap between the two categories.

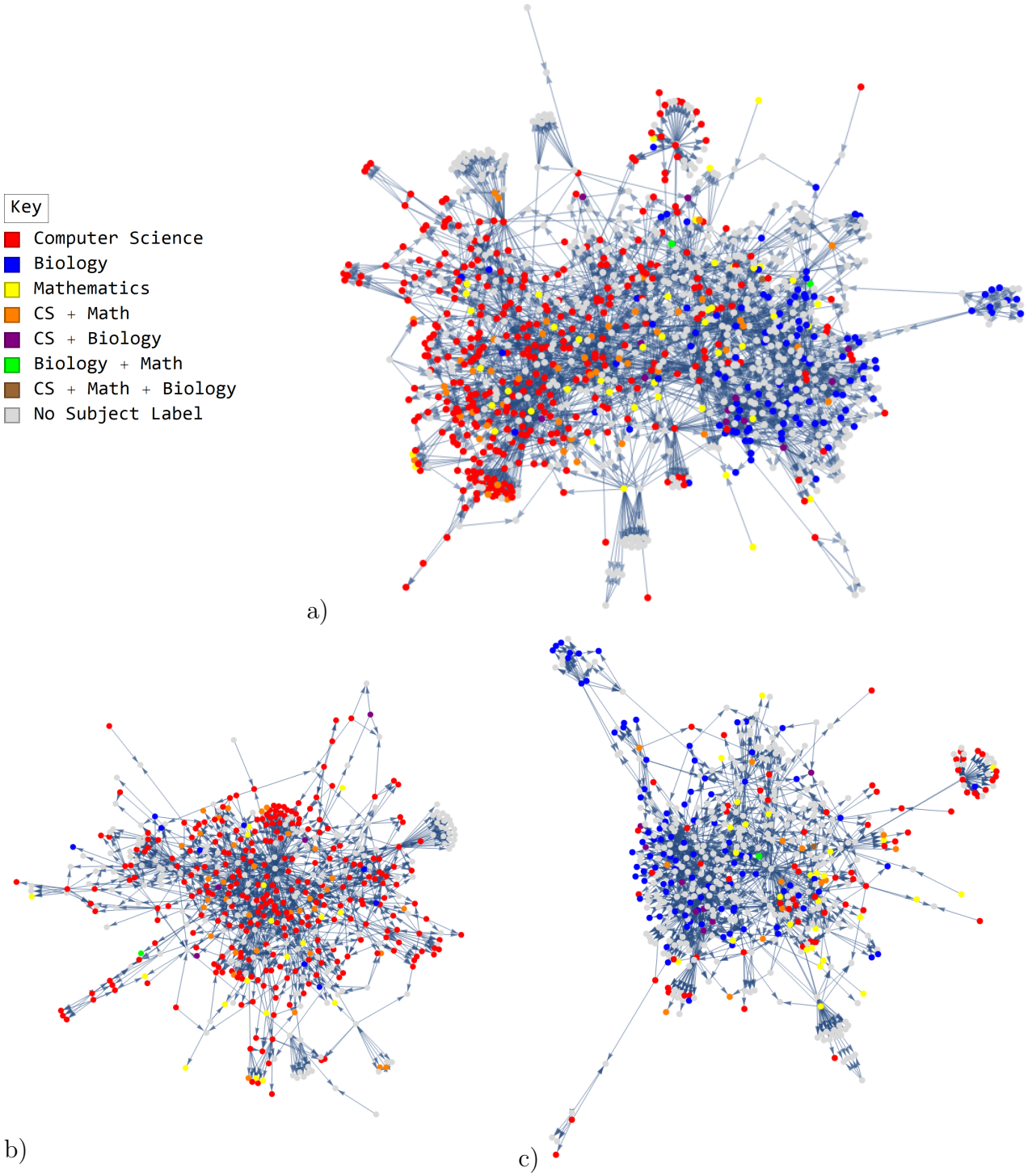


Figure 3.2: a) The pruned network G_p, and b)-c) two halves of its partition $G_p^{(1)}$ and $G_p^{(2)}$, with vertices colored according to their subject label.

	G	G_p	$G_p^{(1)}$	$G_p^{(2)}$
Total vertices	5793	1062	531	531
Untagged	1922	502	311	191
Tagged	3871	560	220	340
CS	2533	405	93	312
Biology	984	122	108	14
Math	787	97	44	53
Both CS and biology	108	13	9	4
Both CS and math	305	49	15	34
Both biology and math	24	3	2	1
All three	4	1	1	0

Table 3.1: Number of vertices tagged as computer science, biology, math, or some combination of these in G, G_p, and the two halves of the partition $G_p^{(1)}$ and $G_p^{(2)}$.

identified as such, and therefore more structurally visible to the partitioning algorithm.

We then count the number of vertices in each color-coded category, as shown in Table 3.1. This confirms what we can intuitively see in Figure 3.2. That is, almost all of the biology papers are found on one side of the partition, the majority of the computer science papers are found on the other side, and the math papers are fairly evenly spread between the two. The number of biology papers is much smaller than the number of computer science papers, which helps explain why there are more untagged papers on the biology side, and why there are significantly more computer science papers on the biology side than there are biology papers on the computer science, both by percentage and by total number.

Color codings for the full network and the subnetwork of high centrality papers can be found in Figures A.3 and A.4, and a table of vertex counts similar to Table 3.1 for the high centrality subnetwork can be found in Table A.1.

3.2.1 Assortativity results. We would like to calculate the assortativity of the network with respect to our subject tagging, to measure the degree to which papers on a certain topic cite other papers on the same topic. It is unclear how to do so, however, since our vertices can belong to multiple categories, while the assortativity algorithm requires categories to be exclusive. To handle this issue, we report results in Table 3.2 for multiple strategies for

	G	G_p
Outdegree	-0.0178	-0.0141
Publication year	0.0067	0.0041
Citation count	0.0006	0.0654
Reference count	0.0193	-0.0061
Tagged with any subject	0.1089	-0.0094
Subject	0.1837	0.0712
Subject is CS	0.2624	0.1529
Subject is biology	0.3354	0.1773
Subject is math	0.0732	0.0164
Subject is CS or biology	0.1500	0.0188
Subject is CS or math	0.2458	0.1256
Subject is biology or math	0.1713	0.0414

Table 3.2: Assortativity of the full and pruned citation networks with respect to various network properties.

dealing with multiple category membership. We can either define category intersections to be their own, separate category, which was the approach for the "Subject" row in Table 3.2, or we can calculate assortativity with respect to whether a vertex is or is not tagged as a certain subject or group of subjects, which was the approach for the rest of Table 3.2.

For non-subject properties, our assortativity values are all very low in absolute value, meaning that vertices are neither more or less likely to cite vertices with similar outdegree, publication year, citation count, or reference counts as themselves.

We do, however notice nontrivial assortativity with respect to several our subject-based properties. The values are much lower than what we observed for the example in Figure 1.7, which had an assortativity of 0.72, but this is not surprising. Many of the papers on each of our topics could not be tagged as such, so the assortativity is not as high as it likely would be with perfect subject tagging. We also would not expect to see as much assortativity in the citation network of an interdisciplinary academic research area as we would in a network of non-academic political books, especially when there is significant overlap between our categories. Survey papers in particular will lower the assortativity as they draw connections between work on a similar topic, but in different disciplines.

We first notice that the assortativity values in G_p are lower than their corresponding

values in all of G. That is, the papers with only one parent, which we have removed in G_p, are more likely to have the same subject tag as their parent than those cited by multiple papers. We also observe that there is very little assortativity with respect to whether a paper's subject is mathematics, which justifies our hypothesis that the assortativity with respect to computer science and biology is noteworthy, and not observed in any subject classification.

Finally, we note that the assortativity with respect to whether a paper is either computer science or biology, or neither, is much lower than with respect to either category on its own, and only somewhat higher than the assortativity with respect to whether a paper is tagged at all. Then the assortativity with respect to whether a paper is computer science or math is only slightly lower than with respect to computer science by itself, while the assortativity with respect to whether a paper is computer science or biology is much lower than with respect to biology by itself. That is, the math category is more highly structurally linked to computer science than biology (which is unsurprising, given the relative sizes of its intersections with each category), and biology is the most structurally distinct category overall.

3.3 How centrality and context inform a better survey

In Section 2.3.1, we introduced a collection of 61 papers which were found to have high centrality either overall or within one of the sides of our partition of the pruned network. Our goal is to frame our presentation around the most important papers in the network, so they form our primary reading list.

To facilitate our reading, we collected and tabulated metadata for the high centrality papers, including their author, year, title, DOI number, whether they are a parent in the network, their rank with respect to each of our centrality metrics overall and within each side of the partition, and their overall rank as discussed in Section 2.3.1. We also considered the subnetwork of our high centrality vertices, which we refer to as G_R, and visually organized them as shown in Figure 3.3 and Figure 3.4 to allow us to see at a glance the context of each

paper in the wider reading list. In Table A.2, we show how many vertices in each paper's neighborhood within G_R fall on either side of the partition, which we can use as a guide to which papers might be highly interdisciplinary either in the references they cite, or the papers they are cited by. Finally, we can use the Mathematica representation of the network to easily calculate how many and which of a paper's references are in the pruned network and on either side of the partition, and check the intersection of the neighborhoods of two or more papers.

At this point, we have a powerful amount of context to guide our reading. We know that there are two main categories of application in our dataset, we know roughly how they inform its structure, and we know which papers are important in each category as well as overall. We know that we are only reading papers which are considered important in some way, we know that they are important within our topic of interest specifically, and we know exactly *how* important they are considered to be in comparison to the rest and why.

As we read, we can easily check a paper's neighbors against those we have already read to place it within the context of the overall shape of the field , and to compare our computational insights into which of the references included are relevant with the author's choices in how to frame them. We can choose to start by reading the papers with high authority centrality, in order to gain an understanding of concepts before building upon them, and follow both year and forward reference information to track the development of the field over time. We can check for connections in the form of cocitations between papers we might expect to be connected based on the ideas they discuss–which is interesting both when the connections are there, and when they are not–and thereby get a sense of what the idea transfer between the two disciplines has been up to this point, which ideas have found crossover and which have not, and why.

Overall, we acquire a global sense of the shape of the field of network similarity, which we can trust to be less biased by our own or other author's perceptions of what is important. At this point, it is easy to choose additional papers to read and refer to as needed throughout

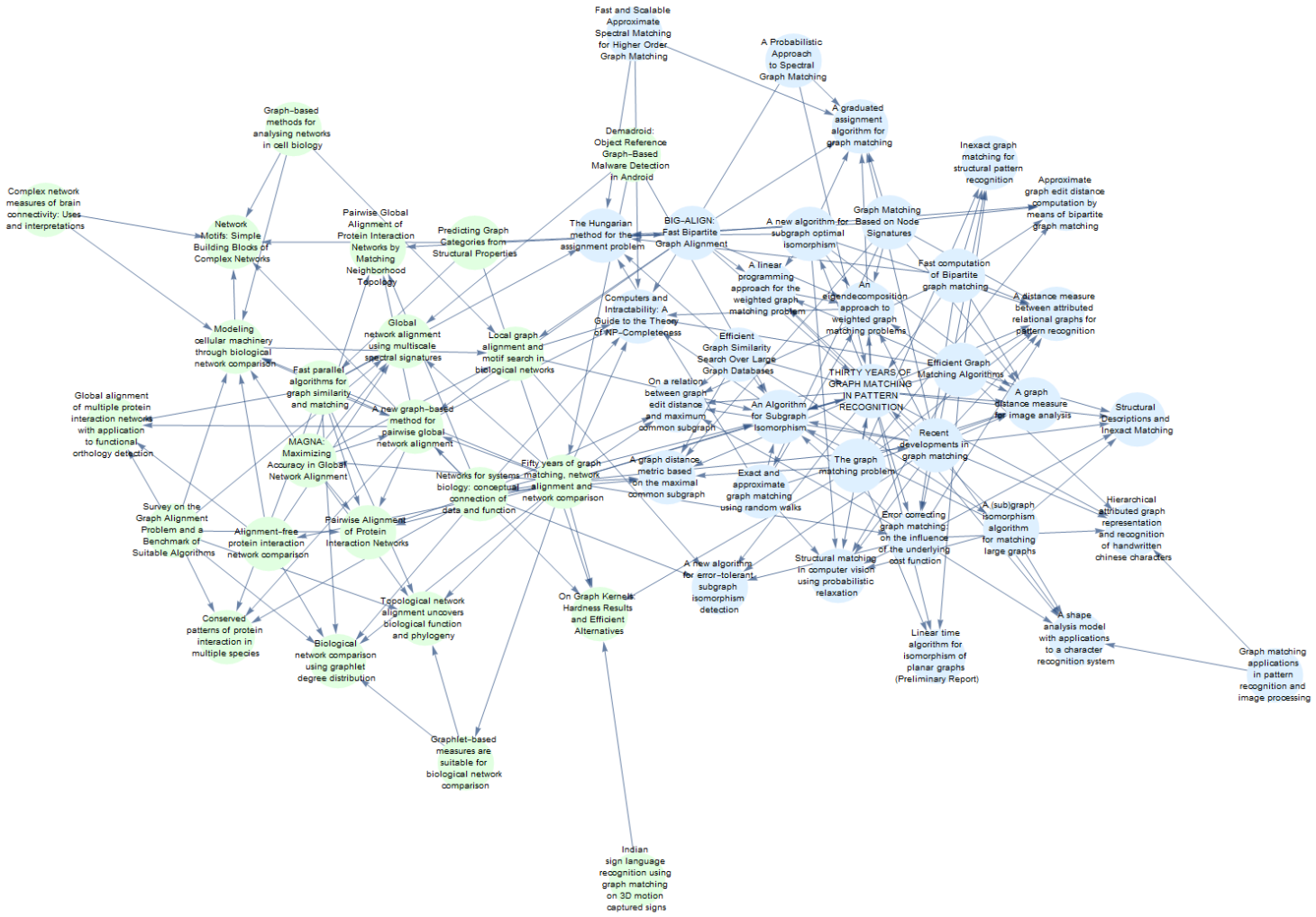


Figure 3.3: The subnetwork S of high centrality papers, as listed in Tables 2.2, 2.3, and 2.4. Green vertices are in group 1 (biology dominated) of the partition of G_p, and blue vertices are in group 2 (CS dominated).

Note: "Unsupervised Domain Adaptation Using Regularized Hyper-Graph Matching" is not in the connected component and is not displayed.

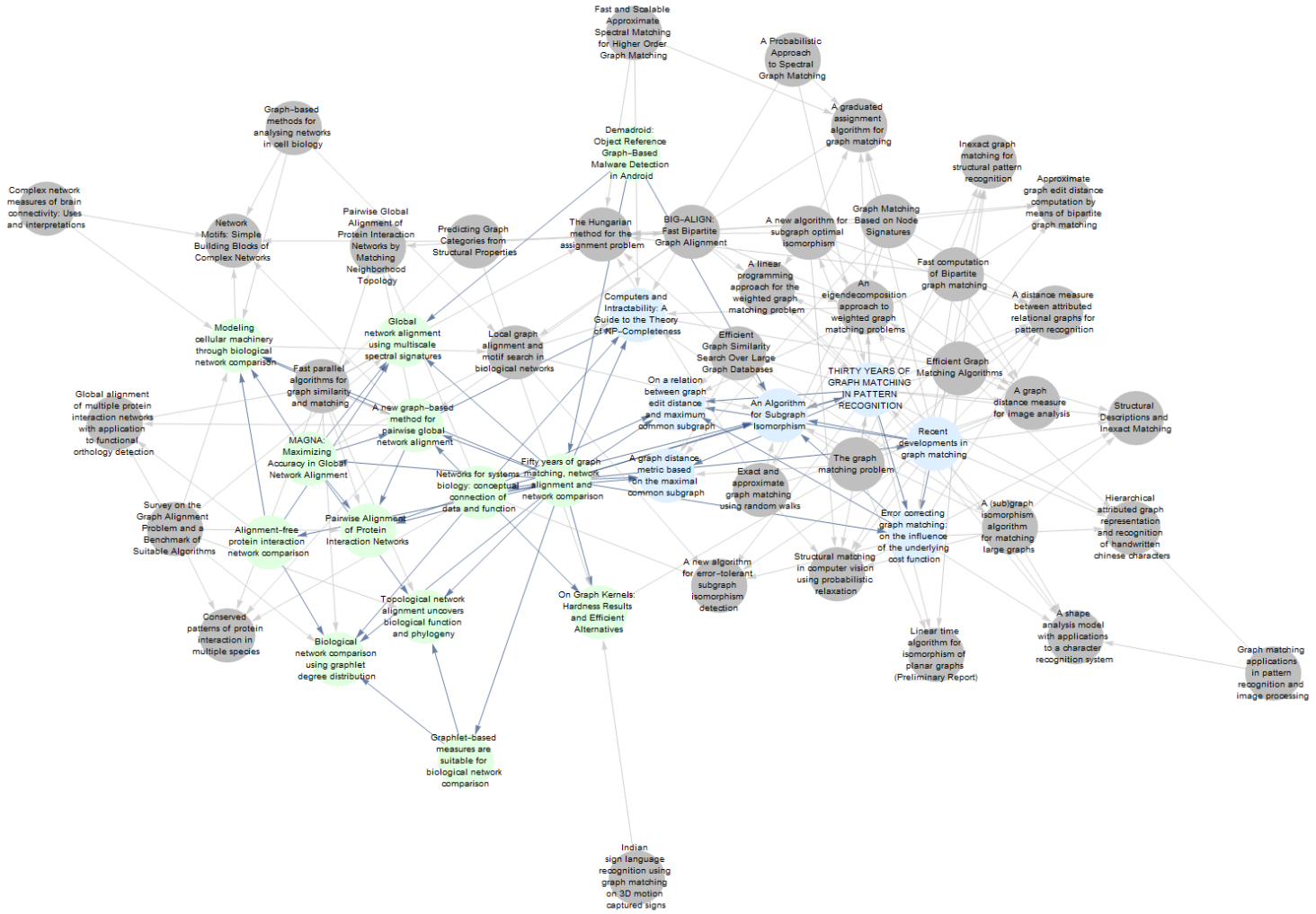


Figure 3.4: The subnetwork S of high centrality vertices, highlighting the neighborhood of "Fifty years of graph matching, network alignment, and network comparison" [26].

Note: "Unsupervised Domain Adaptation Using Regularized Hyper-Graph Matching" is not in the connected component and is not displayed.

the reading and writing process, and to use their reference lists to put them in the context of our existing understanding.

As a last remark, we note that our context is based largely on the reading list subnetwork alone–which was formed using an arbitrary cutoff of "top ten papers"–and that the network mathematics involved is quite basic. Even as helpful as we have found this context to be, we have likely only scratched the surface of the possible benefits of this method of exploration.

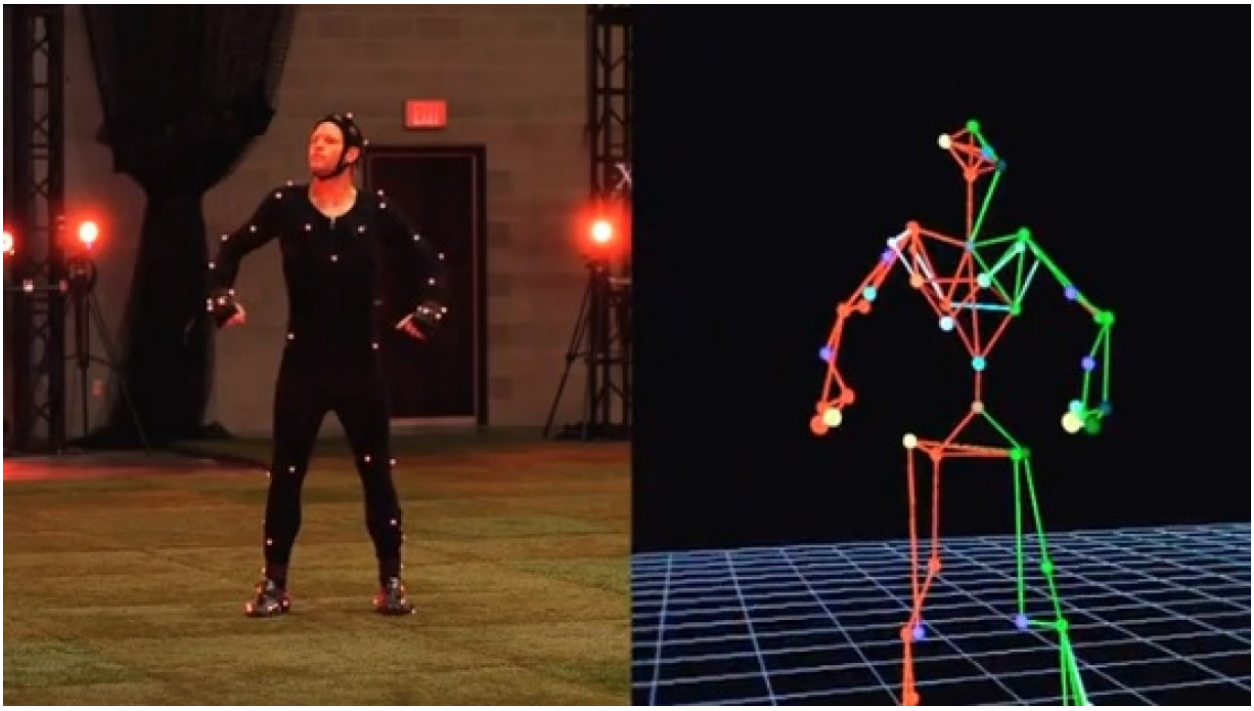
In the following chapters, we outline the motivation and approach for our two main categories of application. We begin with pattern recognition in Chapter 4 and then discuss biology in Chapter 5.

Chapter 4. Pattern Recognition

4.1 Motivation

The complex, combinatorial nature of graphs makes them computationally very difficult to work with, but it also makes them an incredibly powerful data structure for the representation of various objects and concepts. They are particularly useful in computer vision, where we would often like to recognize certain objects in an image or across images that ~~might~~ seem ~~very~~ different at the pixel level as a result of things like angles, lighting, and image scaling. Since graphs are invariant under positional changes including translations, rotations, and mirroring, they are well suited for this task.

Applications in the area of computer vision include optical character recognition [51, 65], biometric identification [38, 20] ~~and~~ medical diagnostics [73], and 3D object recognition [14]. In 2018, work has been published in the computer vision-related areas of Indian sign language recognition [48, 47], spotting subgraphs (e.g. certain characters) in comic book images [49], and stacking MRI image slices [15]. A timeline with more comprehensive counts of papers appearing in various applicative areas in pattern recognition through 2002 can be found in [17].



Source: http://ultimatefifa.com/2012/fifa-13-motion-capture-session/

Figure 4.1: A graph-based representation of a human body, with vertices corresponding to the markers on a motion capture suit.

In computer vision applications, as well as for pattern recognition in general, we create a graph representation for an image by decomposing it into parts and using edges to represent the relationships between these components. For example, we can describe a person using the relationships between various body parts–head, shoulders, knees, toes, and so on. This is the idea behind motion capture, as illustrated in Figure 4.1. After we have a graph representation of the objects we would like to compare, the problem of recognition, and in particular of database search, is reduced to a graph matching problem. We must compare the input graph for an unknown object to our database of model graphs to determine which is the most similar.

4.2 Defining graph matching

In the literature, the term "graph matching" is used far more often than it is explicitly defined. When a definition is given, it is usually tailored to the purposes of a particular

author, and specific to a certain *type* of graph matching; i.e. exact, inexact, error-correcting, bipartite, and so on. The distinctions between these can be subtle, and are typically only explicitly addressed in survey papers. Furthermore, we sometimes address the question of finding a matching *in* a graph [84], which is different from but still related to the problem of *graph matching*, in which we want to find a mapping *between* two graphs. And finally, ~~our Google Scholar results return~~ there are a significant amount of papers about **elastic graph matching**, which is widely used in pattern recognition but is not in fact a form of graph matching [16]. Clearly a good taxonomy is needed.

don't bold this unless you plan to define it.

In this section, we give an overview of graph matching-related terms and summarize their distinctions.

4.2.1 Preliminary definitions. Graph isomorphism is the strictest form of graph matching and a natural place to begin our discussion. We therefore give definitions for graph isomorphism and the two main relevant generalizations of the graph isomorphism problem, as well as the temporal complexity-related terms necessary to compare their computational difficulty.

You must be forcing the break here → don't do it!

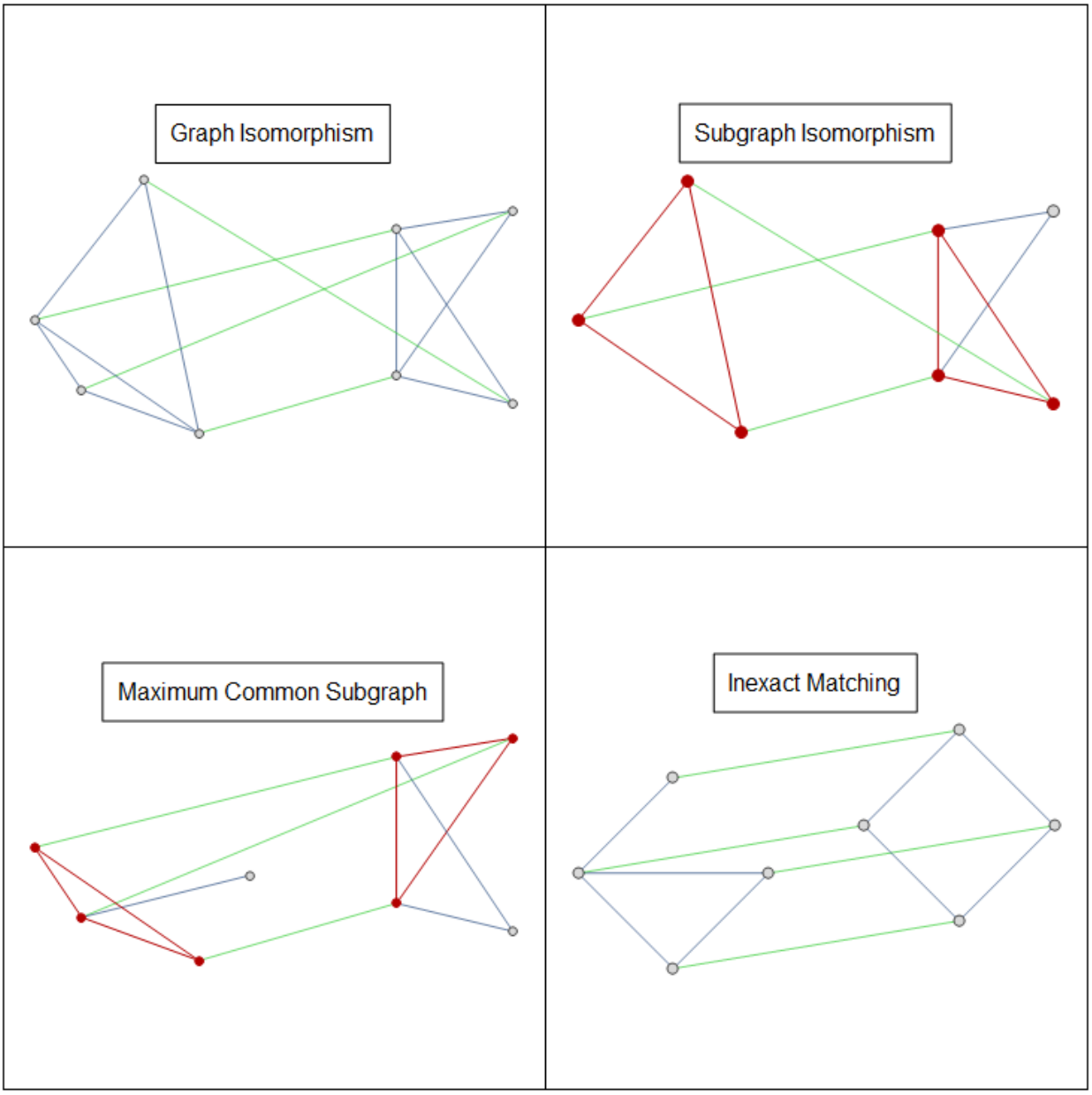


Figure 4.2: A visual summary of the distinctions between graph isomorphism, subgraph isomorphism, maximum common subgraph, and inexact matching.

A decision problem (i.e., one which can be posed as a yes or no question), of which the graph isomorphism problem is an example, is in **NP** or **nondeterministic polynomial time** if the instances where the answer is "yes" can be verified or rejected in polynomial time [35, 86]. It is **NP-hard** if it is "at least as difficult as every other NP problem"; that is, every problem which is in NP can be reduced to it in polynomial time. An NP-hard problem

does not necessarily have to be in NP itself [35, 87]. If a decision problem is both NP and NP-hard, it is **NP-complete** [35, 85].

An **induced subgraph** of a graph is a graph formed from a subset of vertices in the larger graph, and all the edges between them [83]. By contrast, a **subgraph** is simply a graph formed from a subset of the vertices and edges in the larger graph [81].

A **graph isomorphism** is a bijective mapping between the vertices of two graphs of the same size, which is **edge-preserving**; that is that is, if two vertices in the first graph are connected by an edge, they are mapped to two vertices in the second graph which are also connected by an edge [17]. The decision problem of determining whether two graphs are isomorphic is neither known to be in NP nor known to be solvable in polynomial time [82].

A **subgraph isomorphism** is an edge-preserving injective mapping from the vertices of a smaller graph to the vertices of a larger graph. That is, there is an isomorphism between the smaller graph and some induced subgraph of the larger [17]. The decision problem of determining whether a graph contains a subgraph which is isomorphic to some smaller graph is known to be NP-complete [88].

Finally, a **maximum common induced subgraph** (MCS) of two graphs is a graph which is an induced subgraph of both, and has as many vertices as possible [80]. Formulating the MCS problem as a graph matching problem can be done by defining the metric

$$d(G_1, G_2) = 1 - \frac{|MCS(G_1, G_2)|}{\max\{|G_1|, |G_2|\}},$$

where $|G|$ is the number of vertices in G [11, 8].

4.2.2 Exact and inexact matching. We define a graph matching method to be **exact** if it seeks to find a mapping between the vertices of two graphs which is edge preserving. Exact matching is also sometimes defined by whether a method seeks a *boolean* evaluation of the similarity of two graphs [50, 26]. For graph and subgraph isomorphism, this characterization is equivalent; either they are isomorphic/there is a subgraph in the larger which is isomorphic

	Graph isomorphism	Subgraph isomorphism	Maximum common induced subgraph
G_1 and G_2 must have the same number of vertices	X		
Mapping must include all vertices of either G_1 or G_2	X	X	
Mapping must be edge-preserving	X	X	X
NP-complete	Unknown	X	X*

*The associated decision problem of determining whether G_1 and G_2 have a common induced subgraph with at least k vertices is NP-complete, but the problem of finding the maximum common induced subgraph (as required for graph matching) is NP-hard [80].

Table 4.1: A summary of exact graph matching problem formulations.

to the smaller, or they are not. Since the maximum common subgraph problem is edge preserving, we consider it in this work to be an exact matching problem. However, it does not seek a boolean evaluation, and it is therefore sometimes considered to be an inexact matching problem [50].

In **inexact matching**, we allow mappings which are not edge-preserving, which allows us to compensate for the inherent variability of the data in an application, as well as the noise and randomness introduced by the process of constructing graph representations of that data. Instead of matchings between vertices being forbidden if edge-preservation requirements are unsatisfied, they are simply penalized in some way. That is, we seek to find a matching that minimizes the sum of this penalty cost. Instead of returning a value in $\{0, 1\}$, we return a value in $[0, 1]$ measuring the similarity or dissimilarity between two graphs[1].

Inexact matching algorithms which are based on an explicit cost function or edit distance are often called **error tolerant** or **error correcting**.

Optimal and approximate algorithms. Generally, the problem formulations used for inexact matching seek to minimize some nonnegative cost function which can be defined to theoretically be zero for two graphs which are isomorphic. An **optimal** algorithm, that is,

[1]Returning 1 for an isomorphism is analagous to a boolean evaluation and would be considered a *similarity* measure. Most algorithms for inexact matching seek to minimize some function, so they would return 0 for an isomorphism and are therefore considered *dissimilarity* measures.

	Edge preserving?	Result in?	Mapping seeking?	Optimal?	Complexity
Graph isomorphism	Yes	{0,1}	Yes	Yes	Likely between P and NP
Subgraph isomorphism	Yes	{0,1}	Yes	Yes	NP-complete
MCS computation	Yes	[0,1]	Yes	Yes	NP-hard
Edit distances (exact)	No	[0,1]	No	Yes	Generally exponential
Edit distances (approximate)	No	[0,1]	No	No	Generally polynomial
Other inexact formulations	No	[0,1]	Sometimes	No*	Generally polynomial

Table 4.2: Summary of the distinctions between exact and inexact graph matching styles. *It's possible that optimal methods for inexact matching exist, but we did not observe any. If they exist, they are almost certainly not polynomial time.

one which is guaranteed to find the global minimum of this function, is guaranteed to find an isomorphism if it exists, while still handling the problem of graph variability. However, this comes at the cost of making optimal algorithms for inexact matching significantly more expensive than their exact counterparts [17].

Most inexact matching algorithms are therefore **approximate** or **suboptimal**. They only find a local minimum of the cost function, which may or may not be close to the true minimum (which may or may not be acceptable in a certain application), but in return are much less expensive to calculate, usually polynomial time [17].

Mapping-seeking and non-mapping-seeking algorithms. Finally, we can draw the distinction of whether the algorithm seeks primarily to find a mapping between vertices (and returns a result in $\{0, 1\}$ or $[0, 1]$ as a byproduct), or whether it does not. All exact formulations seek a mapping, and many inexact formulations do as well. Mapping-seeking inexact matching is more commonly referred to as **alignment**, and is one of two overwhelmingly dominant comparison strategies we observed in biology applications. Alignment is discussed in more detail in Chapter 5.

4.2.3 Graph kernels and embeddings. "The Graph Matching Problem" [50], published in 2012, claims that there are three main approaches to the inexact graph matching problem: edit distances, graph kernels, and graph embeddings. We did not observe this to be the case in our reading, but we still give a brief introduction to graph kernels and embeddings, and discuss our observations.

Graph **embeddings** are a general strategy of mapping a graph into some high-dimensional feature space, and performing comparisons there [25]. For example, we could identify a graph with a vector in $\mathbb{R}^n$ containing the seven statistics reported in Table 2.1, or the eigenvalues of its adjacency matrix, and compare them using Euclidean distance. Mapping a graph into Euclidean space certainly makes comparison easier, but it is not obvious how to create a mapping that preserves graph properties in a sensible way.

Graph **kernels** are a special kind of graph embeddings, in which we have a continuous map $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$, where $\mathcal{G}$ is the space of all possible graphs, such that k is symmetric and positive definite or semidefinite [50]. Creating a kernel for graphs would allow us to take advantage of the techniques and theory of general kernel methods, but it has been shown that computing a strictly positive definite graph kernel is at least as hard as solving the graph isomorphism problem [33]. We suspect that the amount of work involved in creating a graph kernel with enough desirable properties to take advantage of kernel methods is prohibitive enough in many cases to make this an impractical strategy.

We note that the strategy of using the **graphlet degree distribution** and other local and global graph statistics, which we discuss in Chapter 5, is a form of embedding. Furthermore, the graph kernel strategies described in the references of [50] seem to follow the assignment problem-style approach of calculating some sort of similarity notion between pairs of vertices in two graphs, and using that matrix to create the desired alignment or kernel.

We therefore consider the strategies of graph kernels and graph embeddings to be within the families of other categories which we describe in this work, rather than mainstream

approaches in their own right.

4.3 Exact matching and graph edit distance

The field of graph matching is large and well-established, and we cannot hope to give a full overview of all existing techniques without sacrificing our focus on remaining accessible to the relative novice. If the reader is interested in a more comprehensive investigation, the definitive source on graph matching developments through 2004 is "Thirty Years of Graph Matching In Pattern Recognition" [17]. Two of its authors collaborated with various others on a similar survey in 2014 covering the ten years since the prior survey's publication [29], and in June of 2018 published a large-scale performance comparison of graph matching algorithms on huge graphs [13] that may also be of interest.

We roughly partition the field into three main approaches:

(i) Exact matching methods, which are primarily based around some kind of pruning of the search space

(ii) Edit distance-based methods for optimal inexact matching, and

(iii) Continuous optimization-based methods for inexact matching.

We present optimization methods in their own section, as they are relevant to our upcoming discussion of systems biology network comparison methods in Chapter 5, and in this section aim to introduce the concept of search space pruning (which is the most dominant approach for exact matching), and the concept of a graph edit path and its corresponding graph edit distance. Our presentation of edit distances is primarily inspired by [50] and [64].

4.3.1 Search space pruning. Most algorithms for exact graph matching are based on some form of tree search with backtracking [17]. The process is analagous to solving a grid-based logic puzzle. We represent all possible matching options in a grid format, and then rule out infeasible possibilities based on clues or heuristics about the problem. When we get to

the point where our clues can no longer rule out any further possibilities, we must arbitrarily choose from among the remaining possible options for a certain item and follow through the correspondingly ruled out possibilities until we either complete the puzzle or reach a state where there are no possible solutions remaining. In the latter case, we backtrack, rule out our initial arbitrary choice, and try other possible options for the same certain item until we either find a solution or exhaust all possible choices.

The seminal algorithm for exact matching is found in Ullmann's 1976 paper "An Algorithm for Subgraph Isomorphism" [76], and is applicable to both graph and subgraph isomorphism. We assume two graphs g_1 and g_2 with vertex counts m and n, respectively, and assume without loss of generality that $m \leq n$. This allows us to represent all matching candidate possibilities in a $m \times n$ matrix of zeros and ones.

The Ullmann algorithm uses two principles to rule out matching possibilities:

(i) In a subgraph isomorphism, a vertex in g_1 can only be mapped to a vertex in g_2 with the same or higher degree. This is used to rule out possibilities initially.

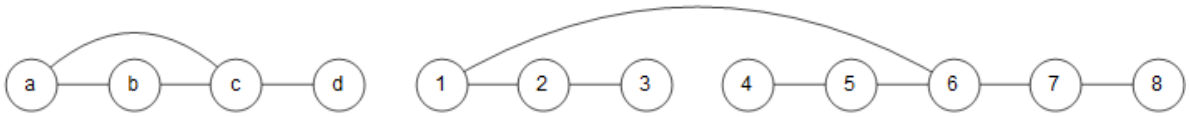
In Example 4.1, degree comparison is able to reduce the number of possible matchings from $8^4 = 4096$ to $5^*5^*1^*8 = 200$, a drastic improvement found at a cost of at most mn operations (comparing the degree of each vertex in g_1 against the degree of each vertex in g_2).

(ii) For any feasible matching candidate $v_2 \in g_2$ for $v_1 \in g_1$, the neighbors of v_1 must each have a feasible matching candidate among the neighbors of v_2. Testing this is called the **refinement** procedure, and it forms the heart of the algorithm.

In Example 4.1, after a single stage of the refinement process and before we begin backtracking, we have reduced the number of possible matchings down to $3^*3^*1^*3 = 27$.

Example 4.1. As Ullmann's presentation of his own algorithm is very detail-oriented and does not seek to give a broad intuition for the method, to illustrate the method we follow

an example found on StackOverflow [60]. We have two graphs g_1 and g_2, as shown, and we want to determine if a subgraph isomorphism exists between them.



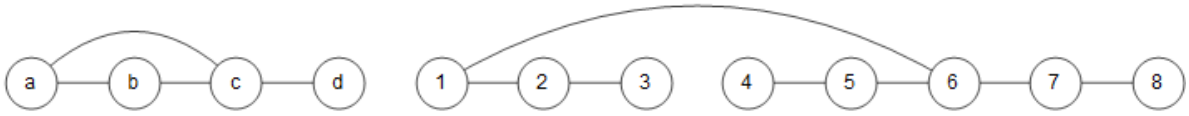
g_1 g_2

First, we use degree comparison to determine the initial candidates for mapping vertices in g_1 to vertices in g_2. Vertex d has degree 1, so it can be mapped to anything in g_2. Vertices a and b have degree 2, so they cannot be mapped to vertices 3, 4, or 8 in g_2, as these vertices have degree 1. Finally, vertex c has degree 3, so it can only be mapped to vertex 6.

	1	2	3	4	5	6	7	8
a	1	1	0	0	1	1	1	0
b	1	1	0	0	1	1	1	0
c	0	0	0	0	0	1	0	0
d	1	1	1	1	1	1	1	1

Candidate mapping pairs which satisfy degree requirements.

Next, we begin the refinement procedure. We show two cases of the refinement procedure for the candidates of vertex a in g_1: one where the candidate is valid, and another where it is ruled out.



g_1 g_2

	1	2	3	4	5	6	7	8
a	1	1	0	0	1	1	1	0
b	1	1	0	0	1	1	1	0
c	0	0	0	0	0	1	0	0
d	1	1	1	1	1	1	1	1

Vertex 1 is a suitable candidate for vertex a.

	1	2	3	4	5	6	7	8
a	1	1	0	0	1	1	1	0
b	1	1	0	0	1	1	1	0
c	0	0	0	0	0	1	0	0
d	1	1	1	1	1	1	1	1

Vertex 2 is not a suitable candidate for vertex a.

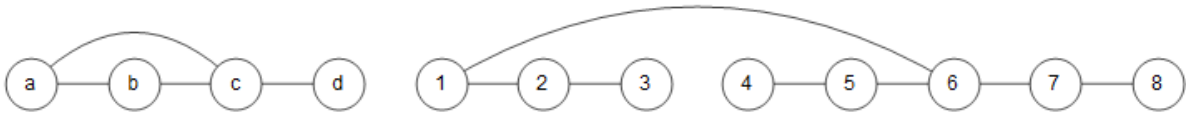
On the left, we consider vertex 1 in g_2 as a candidate for vertex a in g_1. We highlight the rows corresponding to a's neighbors, and the columns corresponding to 1's neighbors. Each neighbor of a must have a candidate among the neighbors of 1; i.e., there must be a 1 somewhere in the intersections of the highlighted columns with each highlighted row. Since this is the case on the left, 1 remains a candidate for a. On the right, however, we find that vertex 2 is not a valid candidate for a. While there is a candidate for b among the neighbors of 2, there is not a candidate for c among the neighbors of 2.

After performing the refinement process for all candidate pairings, the remaining candidates for each vertex in g_1 are as shown below. At this point, it is time to begin backtracking.

	1	2	3	4	5	6	7	8
a	1	0	0	0	1	0	1	0
b	1	0	0	0	1	0	1	0
c	0	0	0	0	0	1	0	0
d	1	0	0	0	1	0	1	0

Candidate mapping pairs after the initial refinement procedure.

For the backtracking procedure, we try mapping a vertex to each of its candidates in turn. At each stage, if we cannot find any viable candidates for a vertex among the neighbors of the candidate in question, we backtrack and try again. The algorithm stops when we either find a subgraph isomorphism, or eliminate all candidates for a vertex.



g_1 g_2

	1	2	3	4	5	6	7	8
a	✗	0	0	0	1	0	1	0
b	0	0	0	0	1	0	1	0
c	0	0	0	0	0	1	0	0
d	0	0	0	0	1	0	1	0

Try mapping a to 1. There is no viable candidate for b among the neighbors of 1, so we backtrack and try again.

	1	2	3	4	5	6	7	8
a	0	0	0	0	✗	0	1	0
b	1	0	0	0	0	0	1	0
c	0	0	0	0	0	1	0	0
d	1	0	0	0	0	0	1	0

Try mapping a to 5. There is no viable candidate for b among the neighbors of 5, so we backtrack and try again.

	1	2	3	4	5	6	7	8
a	0	0	0	0	0	0	✗	0
b	1	0	0	0	1	0	0	0
c	0	0	0	0	0	1	0	0
d	1	0	0	0	1	0	0	0

Try mapping a to 7. There is no viable candidate for b among the neighbors of 7, and no more candidates for a, so we stop.

We cannot find a suitable candidate for b among the neighbors of any candidate of a, so there is no subgraph isomorphism between g_1 and g_2.

4.3.2 Graph edit distance. One way to measure the distance between two objects is to measure how much work it takes to turn the first into the second, and take the length of the **edit path**. For example, in Figure 4.3, we find an edit path of length 4 between "cat" and "rat". However, this should clearly not be the edit *distance* between "cat" and "rat", as we can transform one into the other with a single substitution. The **edit distance** between two objects is therefore the *minimum* over the lengths of all possible edit paths between them.

cat	→	c*a*rt	Insertion
c*a*rt	→	*d*art	Substitution
*d*art	→	art	Deletion
*ar*t	→	*ra*t	Transposition

Figure 4.3: Edit operations for strings.

For graphs, the relevant edit operations are **vertex substitution**, **vertex insertion**, **vertex deletion**, **edge insertion**, and **edge deletion**, as illustrated in Figure 4.4. Instead of simply taking the length of the edit path, however, each of these operations is associated with some nonnegative cost function $c(u, v) \in \mathbb{R}^+$ (the "penalty" mentioned in Section 4.2.2) which avoids rewarding unnecessary edit operations by satisfying the inequality

$$c(u, w) \leq c(u, v) + c(v, w),$$

where u, v, and w are vertices or edges, or sometimes null vertices/edges in the case of inser-

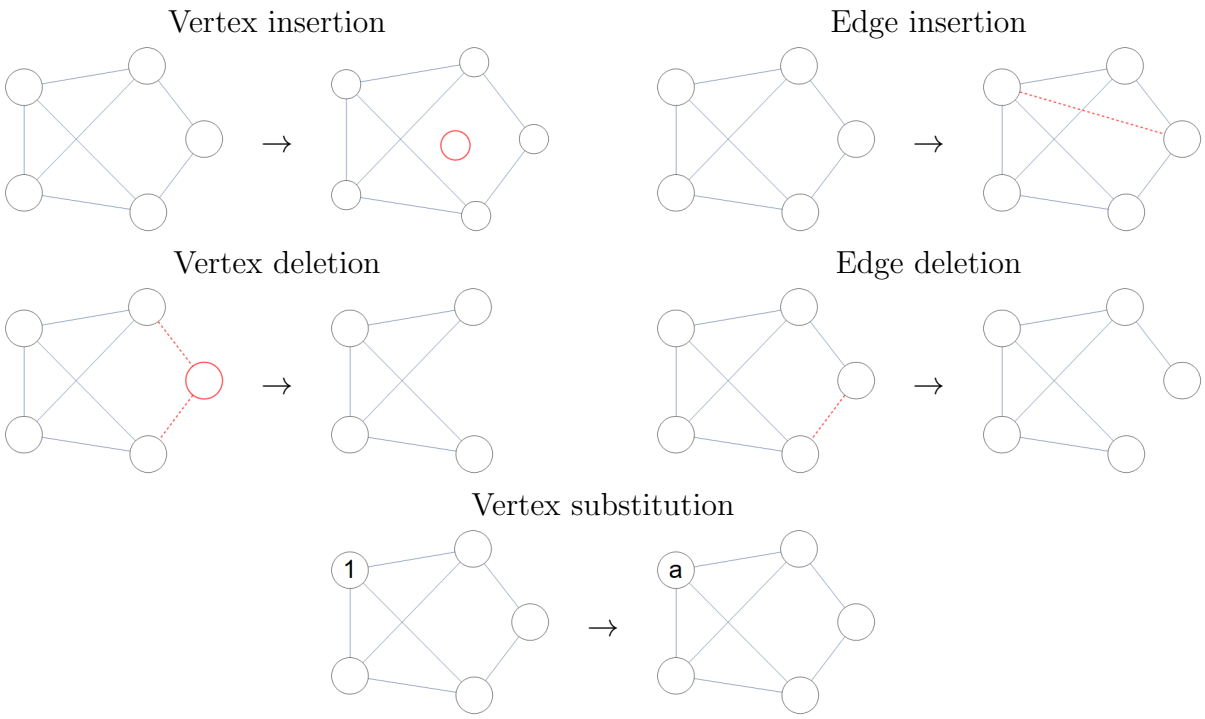


Figure 4.4: Edit operations for graphs.

tion and deletion. We also assume that the cost of deleting a vertex with edges is equivalent to that of first deleting each of its edges and then deleting the resultant neighborless vertex.

The edit distance is then the total cost of all operations involved in an edit path, and it critically depends on the costs of the underlying edit operations [11]. This can be helpful in some cases, as it allows us to easily tweak parameters in our notion of similarity, but it is also sometimes desirable to avoid this dependence on the cost function. This is one motivation for the formulation of inexact graph matching as a continuous optimization problem, which we discuss in the next section.

Finally, we note that it was shown by Bunke in 1999 that the graph isomorphism, subgraph isomorphism, and maximum common subgraph problems are all special cases of the problem of calculating the graph edit distance under certain cost functions [9].

4.4 Suboptimal methods for inexact matching

We noted previously that optimal methods for inexact graph matching tend to be very expensive, and therefore only suitable for graphs of small size. To address this issue, Riesen and Bunke in 2008 introduced an algorithm for approximating the graph edit distance in a substantially faster way [64], which Serratosa published an improved variant of in 2014 [69]. This is not the only suboptimal inexact matching method in existence with the goal of suboptimally calculating a graph edit distance, but it provides an interesting connection between the seemingly radically different strategies of search space pruning and of casting the problem as one of continuous optimization.

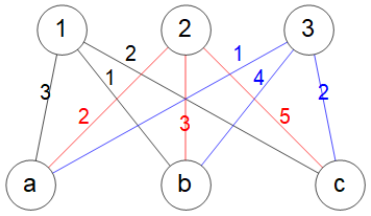
4.4.1 The assignment problem. The key to this connection is the idea of the assignment problem. Instead of searching the space of possible edit paths to find the graph edit distance, we *approximate* the graph edit distance as a matrix optimization problem. The following definition is due to Riesen and Bunke [64]:

Definition 4.2. Consider two sets A and B, each of cardinality n, together with an $n \times n$ cost matrix C of real numbers, where the matrix elements $c_{i,j}$ correspond to the cost of assigning the i-th element of A to the j-th element of B. The **assignment problem** is that of finding a permutation $p = \{p_1, \ldots, p_n\}$ of the integers $\{1, 2, \ldots, n\}$ which minimizes $\sum_{i=1}^{n} c_{i,p_i}$.

A brute force algorithm for the assignment problem would require a $O(n!)$ time complexity, which is obviously unreasonable. Instead, we can use the **Hungarian method**. This algorithm is originally due to Kuhn in 1955 [46] and solves the problem in maximum time $O(n^3)$ by transforming the original cost matrix into an equivalent matrix with n independent zero elements[2] which correspond to the optimal assignment pairs. The version of the algorithm described in [64] is a refinement of the original Hungarian algorithm published by Munkres in 1957 [55].

[2]Independent meaning that they are in distinct rows and columns.

$$C = \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 3 & 2 & 1 \\ 1 & 3 & 4 \\ 2 & 5 & 2 \end{pmatrix} \end{array} \quad \Leftrightarrow$$



$$A = \{a, b, c\}, B = \{1, 2, 3\}$$

Figure 4.5: Reformulating the assignment problem as that of finding an optimal matching in a bipartite graph. The edges and their weight labels in the bipartite graph are colored to make it easier to see which weights belong to which edges.

Relationship to the bipartite graph matching problem. As noted in Section 4.2, we sometimes must address the question of finding a matching *in* a graph. This is defined as a set of edges without common vertices. It is straightforward to reformulate the assignment problem as one of finding an optimal matching within a **bipartite graph**, that is, a graph whose vertices can be divided into two disjoint independent sets such that no edges run between vertices of the same type. If A and B are two sets of cardinality n as in the assignment problem, the elements of A form one vertex group, the elements of B form the other, and we define the edge weight between the i-th element of A and the j-th element of B to be the cost of that assignment, as shown in Figure 4.5. The assignment problem is therefore also referred to as the **bipartite graph matching problem**.

Relationship to graph edit distance. To connect the assignment problem to graph edit distance computation, we define a cost matrix C such that each $c_{i,j}$ entry corresponds to the cost of substituting the i-th vertex in our source graph to the j-th vertex in our target graph [64]. We can generalize this approach further to handle graphs with different numbers of vertices by using a modified version of the Hungarian method which applies to rectangular matrices [6] by considering vertex insertions and deletions as well as substitutions.

In this modified version of the Hungarian method, the resulting cost matrix (again, definition due to Riesen and Bunke [64]) becomes

$$C = \left[\begin{array}{cccc|cccc} c_{1,1} & c_{1,2} & \dots & c_{1,m} & c_{1,-} & \infty & \dots & \infty \\ c_{2,1} & c_{2,2} & \dots & c_{2,m} & \infty & c_{2,-} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \infty \\ c_{n,1} & c_{n,2} & \dots & c_{n,m} & \infty & \dots & \infty & c_{n,-} \\ \hline c_{-,1} & \infty & \dots & \infty & 0 & 0 & \dots & 0 \\ \infty & c_{-,2} & \ddots & \vdots & 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \infty & \vdots & \ddots & \ddots & 0 \\ \infty & \dots & \infty & c_{-,m} & 0 & \dots & 0 & 0 \end{array}\right],$$

/noindent

← where n is the number of vertices in the source graph, m the number of vertices in the target, and a $-$ is used to represent null values. The upper left corner of this matrix represents the cost of vertex substitutions, and the bottom left and top right corners represent the costs of vertex insertions and deletions. Since each vertex can be inserted or deleted at most once, the off-diagonal elements of these are set to infinity. Finally, since substitutions of null values should not impose any costs, the bottom right corner of C is set to zero.

This is only a rough approximation of the true edit distance, as it does not consider any information about the costs of edge transformations. We can improve the approximation by adding the minimum sum of edge edit operation costs implied by a vertex substitution to that substitution's entry in the cost matrix, but we still have a suboptimal solution for the graph edit distance problem, even though the assignment problem can be solved optimally in a reasonable amount of time.

Other suboptimal graph matching methods using the assignment problem. Approximating the graph edit distance is far from the only graph matching strategy which is based around the assignment problem. Instead of a cost matrix based around the cost function of a graph edit distance measure, we can incorporate other measures of similarity or affinity between vertices. The advantage of this approach is that we can incorporate both topological[3] and external notions of similarity into our definition. However, this comes

[3]That is, information derived directly from the structure of a network.

at the cost of relying on heuristic notions of similarity, rather than directly incorporating information about edge preservation into our measure of assignment quality.

Techniques which incorporate external information are much more prevalent in biology, and will be discussed further in Chapter 5.

4.4.2 Weighted graph matching vs. the assignment problem. Most of the suboptimal graph matching methods we observed are based around either the assignment problem, or around some formulation of the *weighted graph matching problem*.

Definition 4.3. The **weighted graph matching problem** (WGMP) is typically defined as finding an optimum permutation matrix which minimizes a distance measure between two weighted graphs; generally, if A_G and A_H are the adjacency matrices of these, both $n \times n$, we seek to minimize $||A_G - PA_HP^T||$ with respect to some norm[77, 43, 4] , or minimize some similarly formulated energy function [31]. The specific norm and definition depends on the technique being used to solve the optimization problem.

Weighted graph matching is an inexact graph matching method, and its techniques are generally suboptimal, searching for a *local* minimum of the corresponding continuous optimization problem. There is a wide variety of techniques in use, including linear programming [4], eigendecomposition [77], gradient descent [43], and graduated assignment [31]. Other techniques mentioned in [4, 77, 31] and [17] ~~for which we do not include specific references~~ include Lagrangian relaxation, symmetric polynomial transformation, replicator equations, other spectral methods, neural networks, and genetic algorithms.

The weighted graph matching problem is similar to the assignment problem in that we seek a permutation between the n vertices of two graphs, but unlike the assignment problem, there is no need to define a cost or similarity matrix ahead of time. Instead, we directly measure the quality of a permutation assignment with respect to the structure of a graph, and optimize this quantity directly to find our matching. This means we can avoid relying on the heuristics inherent in any formulation of a cost or similarity matrix, but it also means

we cannot easily incorporate external information into our solution of the problem. Whether weighted graph matching techniques are preferable to assignment problem-based strategies is therefore dependent on the specific problem to be solved.

Chapter 5. Systems Biology

5.1 Motivation

A fundamental goal in biology is to understand complex systems in terms of their interacting components. Network representations of these systems are particularly helpful at the cellular and molecular level, at which we have large-scale, experimentally determined data about the interactions between biomolecules such as proteins, genes, and metabolites. In the past twenty years, there has been an explosion of availability of large-scale interaction data between biomolecules, paralleling the surge of DNA sequence information availability that was facilitated by the Human Genome Project[1]. Sequence information comparison tools have been revolutionary in advancing our understanding of basic biological processes, including our models of evolutionary processes and disease[2], and the comparative analysis of biological networks presents a similarly powerful method for organizing large-scale interaction data into models of cellular signaling and regulatory machinery [71].

In particular, we can use network comparison to address fundamental biological questions such as "Which proteins, protein interactions and groups of interactions are likely to have equivalent functions across species?", "Can we predict new functional information about proteins and interactions that are poorly characterized, based on similarities in their interaction networks?", and "What do these relationships tell us about the evolution of proteins, networks, and whole species?" [72]. Comparison strategies and metrics are also key to developing mathematical models of biological networks which represent their structure in a

[1]https://www.genome.gov/12011239/a-brief-history-of-the-human-genome-project/

[2]https://www.scq.ubc.ca/the-human-genome-project-the-impact-of-genome-sequencing-technology-on-human-health/ (will fix citation later)

meaningful way, which is a key step towards understanding them. In particular, good comparison techniques allow us to model dynamical systems on biological networks [79] (e.g., the spread of infectious diseases), and create appropriate null hypotheses for drawing conclusions about experimental networks[3] [36].

5.2 Comparison to pattern recognition

As mentioned in Chapter 4, we observed two overwhelmingly dominant network comparison strategies in biology applications. We use the term *comparison* strategies and not *matching* or *alignment* strategies because unlike pattern recognition, not all biological applications seek a mapping between two networks. This is a result of the difference between the typical networks with which each field is concerned; we summarize these distinctions in Table 5.1.

In pattern recognition, graphs are primarily a convenient data structure to represent the objects that we would like to compare, and the goal of comparison is typically to find the "closest" object from a large database. Seeking an "isomorphism with tolerance" is a viable strategy, because we typically have a small number of vertices, and we expect to be able to find a close structural match.

By contrast, in biology we work with networks which are much larger, incompletely explored, and which we do not expect to be "almost the same" globally. Mapping-seeking comparison strategies for biology do not seek to give an overall measure of the similarity between two graphs, but to find *regions* which are conserved between two or more networks. To find useful similarity measures between networks as a whole, we must consider other comparison techniques and ideas.

A typical strategy used as an alternative to alignment is to measure the frequencies and/or distributions of relevant *subgraphs* of a large network. We now introduce the concept of graphlets and motifs, as a generalization of the network statistics we discussed in Chapters 1 and 2.

[3]Section 2.3.1 in [36] is a brief and superb discussion of the importance of modeling biological networks.

Graph matching for pattern recognition	**Biological network alignment**	**Alignment-free biological network comparison**
Seeks to recognize objects based on their graph representations	Seeks to find conserved regions between multiple networks	Seeks to compare networks on a global scale
Attempts to find a mapping or near-mapping between graphs	Attempts to find a mapping or near-mapping between networks	Does not attempt to find a mapping
Deals with large numbers of small graphs	Deals with small numbers of large or very large networks	Deals with a variable number of large or very large networks
Expects to find graphs which are highly structurally similar as a whole	Relevant networks for comparison are not expected to be highly structurally similar as a whole	Relevant networks for comparison are not expected to be highly structurally similar as a whole
Graphs are simply a convenient data structure to represent objects we would like to compare	Each vertex and edge independently represents meaningful information about the real world	Each vertex and edge independently represents meaningful information about the real world

Table 5.1: Summary of differences between graph matching for pattern recognition and biological network comparison.

5.3 Network statistics and measures

In Table 2.1, we compared various network statistics for our constructed citation network and its pruned version to two other citation networks and two other random networks. Our goal in doing so was to determine whether the calculated statistics were noteworthy in some way. Without some kind of context, we cannot tell, for example, whether a diameter of ten is above average, remarkably small, or somewhere in between, or if it is typical to have 96% of the vertices in the network contained in the giant component.

That is, we know that *our network statistics are only meaningful when we compare them to the same statistics for other networks.* Network statistics such as vertex count, edge count, and diameter, therefore, can be thought of as a similarity measure between two networks[4].

[4]See [1] for a discussion of how these are relevant to biology specifically.

Network comparison with univariate measures

Network → Something in $\mathbb{R}^n$
Network → Something in $\mathbb{R}^n$
} → Similarity score derived from a metric or aggregation measure on $\mathbb{R}^n$

Examples: Any metric on $\mathbb{R}^n$ applied to number of vertices and edges, mean degree, diameter, connectivity, degree distribution, centrality distributions, local graphlet counts, graphlet degree distributions, etc.

Network comparison with bivariate measures

Network
Network
} → Mapping process of some sort → Similarity measure derived from the mapping in some way

Examples: Graph edit distance, node correctness, edge correctness, induced conserved structure [59], symmetric substructure score [68], MCS-related metrics (i.e. [11])

Figure 5.1: A summary of the distinction between univariate and bivariate measures.

This is generally a much weaker form of similarity than even suboptimal inexact graph matching, but the more important distinction is that these are **univariate** measures. That is, they map from a single network to $\mathbb{R}^n$, in contrast to **bivariate** measures (i.e. graph matching and alignment), which give us a single number representing the similarity between two networks[5]. The distinction between univariate and bivariate measures is illustrated in Figure 5.1. We can treat univariate measures as bivariate by using some sort of metric or other aggregation measure on its output for two different networks, but they have the additional advantage of allowing us to classify networks based on their output, rank them on a spectrum, and so on. This is particularly important for creating random models of networks that share the key properties of the real-world networks we would like to study [36, 79].

[5]In addition to a mapping between them, if we have an algorithm that returns a similarity measure and not just the mapping.

Type	Description	Examples
Global	Single value for an entire network	examples
Local	Value at each vertex in a network	examples

Table 5.2: A summary of the distinction between local and global network statistics.

5.3.1 Local and global network statistics. Network statistics like vertex and edge count, diameter, size of giant component, and assortativity values are all **global**, meaning they give us a single value for the entire graph. The centrality measures we introduced in Section 2.3, in contrast, are **local**, meaning they give us a value at each vertex. Examples of global and local network statistics can be found in Table 5.2. So far, we have seen these values used to rank vertices and choose a small number of them for further analysis, but we can gain insight into the graph as a whole by looking at *distributions* of these values.

The **degree distribution** is the simplest example of this. We count the number of vertices with degree zero, one, two, and so on, and display the results in a histogram, as illustrated in Figure 5.2. For a directed graph, we have distributions for both indegree and outdegree. Further information about degree distributions and their relevance can be found in [56].

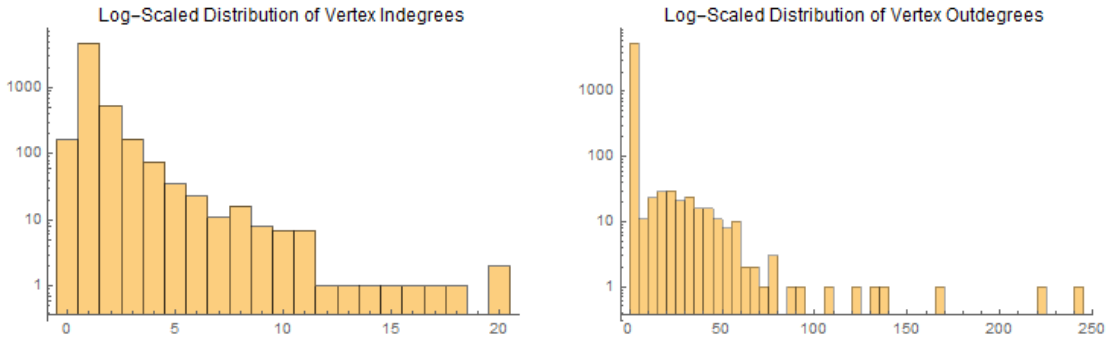


Figure 5.2: Indegree and outdegree distributions for our citation network. As a result of our construction methods, the vast majority of our papers have an outdegree of zero and an indegree of one, so we use a log scale to better observe the spread in the data.

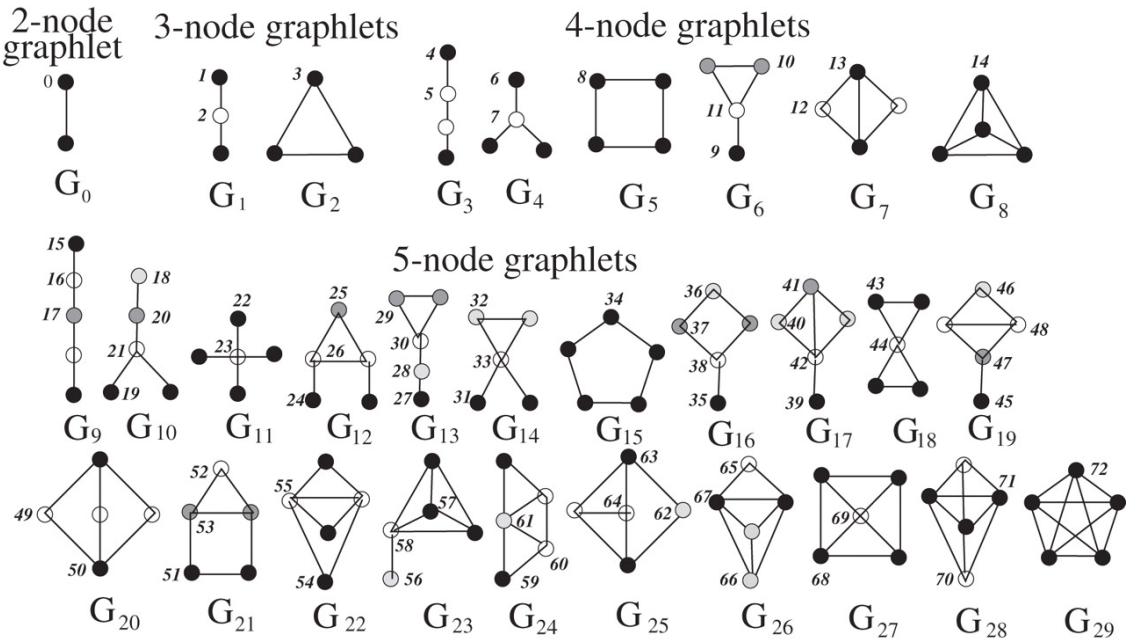


Figure 5.3: The 73 automorphism orbits for the 30 possible graphlets with 2-5 nodes. In each graphlet, vertices belonging to the same automorphism orbit are the same shade. Figure reproduced from [62].

5.4 Graphlets and motifs

Graphlets are small connected non-isomorphic induced subgraphs of a large network [62], introduced by Nataša Pržulj in the mid-2000s to design a new measure of local structural similarity between two networks based on their relative frequency distributions.

First, we recall that the degree distribution measures, for each value of k, the number of vertices of degree k. In other words, for each value of k, it gives the number of vertices touching k edges. We note that a single edge is the only graphlet with two nodes, and call it G_0. The degree distribution can therefore be thought of as measuring how many vertices touch one G_0 graphlet, how many vertices touch two G_0 graphlets, and so on.

We can generalize this idea to larger graphlets, and count how many vertices touch each graphlet $G_0, \ldots, G_{29}$, where $G_0, \ldots, G_{29}$ are defined as in Figure 5.3. However, we notice that for most graphlets larger than two vertices, *which* vertex of the graphlet we touch is topologically relevant; for example, touching the middle node of G_1 is different from touching an end node. This is because the end and middle vertices are in different **automorphism**

orbits. Two vertices are in the same automorphism orbit if they can be mapped to each other in an isomorphism from the graph to itself. Intuitively, this means that we have no way of telling them apart without labeling the graph. If two vertices are in different automorphism orbits, we can tell them apart using their degree, the neighbors of their neighbors, and so on, as we did in our demonstration of the Ullmann subgraph isomorphism algorithm.

In order to define our graphlet degree distributions, then, we count how many vertices touch a certain graphlet *in a specific automorphism orbit*. There are 73 different automorphism orbits for the thirty graphlets with 2-5 nodes, and we therefore obtain 73 **graphlet degree distributions (GDDs)** analagous to (and including) the degree distribution. Since these are based on small local network neighborhoods, they measure the local structure of a graph.

In order to use these distributions for network comparison, we must somehow reduce this large quantity of information to a single measure. Pržulj introduces one possible method in [62] by considering the Euclidean distance between each GDD for two networks, after appropriate scaling and normalization, and then taking the arithmetic or geometric mean over all 73. Other methods are of course possible, and potentially better, depending on the application in question.

5.4.1 Graphlets vs. Motifs. Network **motifs** are similar to graphlets in that they are both small induced subgraphs of large networks. Unlike graphlets, however, the definition of motifs requires these subgraphs to be *statistically overrepresented* in the network [54]; that is, they are patterns of interactions which occur more frequently than you would reasonably expect due to random chance. In a random graph with the same degree sequence(s) as a real network, we are not likely to see connected triangles, for example–but connected triangles appear frequently in real biological networks, associated with feedback or feed-forward loops in transcription and neural networks, clusters in protein interaction networks, and so on [5]. We can generalize further to seek **topological motifs**, which are statistically overrepresented "similar" network sub-patterns.

Such patterns can be used as a first step towards understanding the basic structural elements particular to certain classes of networks [54]; different types of networks contain different types of elementary structures which reflect the underlying processes that generated them, and as discussed in [5], motifs are in fact indicative of biological functions. These methods are a useful way to distinguish patterns of biological function in the topology of molecular interaction networks from random background, but they are not suitable for full-scale comparison of multiple networks [62]. They are sensitive to the choice of random network model used to determine statistical significance, and ignore subnetworks with low or average frequency. These low-frequency subgraphs may still be functionally important, especially if such subnetworks are consistently seen across multiple real networks despite occurring rarely within any individual one. Graphlets are one way to address this issue; alignment strategies are another. We also note that [62] defines graphlets for undirected graphs only, while the motifs discussed in [54] and [5] are primarily directed.

5.4.2 Computational complexity. We already know that subgraph isomorphism is a computationally expensive problem. Exhaustively finding all occurrences of small isomorphic or near-isomorphic subgraphs in a network is infeasible for the gene and protein-protein interaction (PPI) networks of all but small organisms such as *E. coli* [26]. As a result, motif search in large networks (which requires an exhaustive search over an entire *ensemble* of random graphs in order to determine statistical significance) is generally limited to subgraphs of at most five nodes. Graphlet statistics are similarly expensive to compute, and the combinatorially increasing number of possible graphlets for an increasing number of nodes is an additional limitation on the size of graphlets we can reasonably consider. In order to process the interaction networks of higher organisms, various search heuristics and estimation procedures for motifs and graphlet statistics are used.

5.4.3 Netdis. Netdis [3], introduced by Ali et. al. in 2014, is another method for network comparison which is primarily based on counting the occurrences of small subgraphs in a

larger network. For each vertex, they count the number of occurrences of each possible graphlet of 3-5 nodes in a neighborhood of radius two around it. The neighborhoods at each vertex then form an ensemble of subgraph counts, which Netdis centers according to the gold standard for a true protein-protein interaction (PPI) network, as a proxy (in the absence of a suitable random model) for the subgraph counts we would expect to see in a typical PPI network. These centered counts are then combined into an overall statistic.

Ali et. al. use this statistic to correctly separate random graph model types and to build the correct phylogenetic tree of species based on their protein interaction networks, showing that Netdis is a relevant comparison method for large networks. It is also highly tractable; since we only search for subgraphs in a given neighborhood, its computational complexity grows about linearly with the number of vertices in a network, and it is therefore well-suited for full-scale comparison of many large networks.

We note that although Netdis uses graphlet counts, it is not a generalization or special case of the graphlet degree distribution. Not all the graphlets a vertex touches will necessarily be present in its two-step neighborhood, and we can also have graphlets in the two-step neighborhood of a vertex which do not directly touch the vertex itself. As a result, we only need to consider the 29 possible graphlets of 3-5 nodes, rather than their 72 possible automorphism orbits.

5.5 Local and global network alignment

In Chapter 4, we defined network alignment as mapping-seeking inexact matching. Up to this point in our discussion, any mapping found by an algorithm has not been the primary result of interest in a network comparison. When calculating graphlet statistics and network motifs, we do perform graph matching, but the subgraphs involved are so small that whether the matching algorithm is mapping-seeking is not a particularly relevant question; the computational difficulty is a result of the large number of individually trivial matches performed. In pattern recognition, we use inexact matching methods in order to allow for error in the

comparison of slightly larger (up to a hundred nodes at most, typically) graphs that should represent the same or similar objects. There are enough nodes to make an exhaustive search difficult, so the matching method we use matters, but the mapping itself is not typically the primary result of interest.

When comparing large biological networks, however, we frequently seek regions of similarity and dissimilarity in two or more large networks (thousands of nodes and tens of thousands of edges). The mapping between networks is the result of interest, and not simply an intermediate step in finding a low-dimensional similarity score. Similarly to how longer DNA sequences which are conserved across species indicate functional significance and can help classify evolutionary relationships, larger subnetworks of biomolecular interactions which are conserved across species are likely to represent true functional modules [72] and give insight into evolutionary processes [3].

We previously introduced the idea of the assignment problem, where given a matrix whose entries represent the cost of assigning vertices in one network to vertices in another, we seek to find an overall assignment with a minimum total cost. In order to find an alignment, we typically follow a broad outline of

(i) Constructing a cost matrix.

(ii) Using that cost matrix somehow to construct a mapping.

The representative (better adjective?) example of this is of course the Hungarian method-/Munkres' algorithm, as previously discussed. Most of the alignment algorithms we observed follow this same basic template[6], and we frame our discussion of them accordingly. A summary of cost matrix and mapping construction strategies for Hungarian method-style alignment algorithms can be found in Table 5.3.

[6]The term "local alignment" varies in its usage between authors and over time, and local alignment algorithms which fall more under the category of statistical techniques/univariate measures do not appear to use a Hungarian method-style approach. For motifs and graphlets, matching techniques fall more into the category of search space pruning.

5.5.1 Local vs. global alignment. Alignment strategies for biological applications fall into two categories: local alignment, and global alignment. In both cases, we seek to find regions of similarity between networks, and mappings do not need to be defined for every vertex in each network.

In **local alignment**, the goal is to find local regions of isomorphism between the two networks, where each region implies a mapping which is independent of the others. These mappings do not need to be mutually consistent; that is, a vertex can be mapped differently under different regions of the mapping, as illustrated in Figure 5.5. We can choose a locally optimal mapping for each region of similarity, even if this results in overlap. In **global alignment**, by contrast, we must define a single mapping across all parts of the input, even

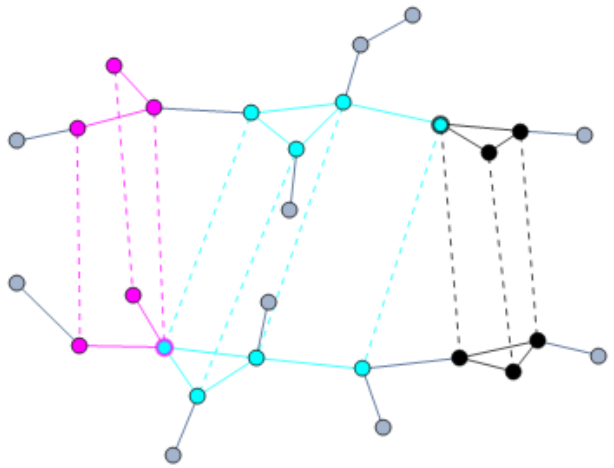


Figure 5.4: Local alignment of a network. Vertices may be used for multiple "pieces" of the overall mapping, i.e. the mapping is not required to be one-to-one.

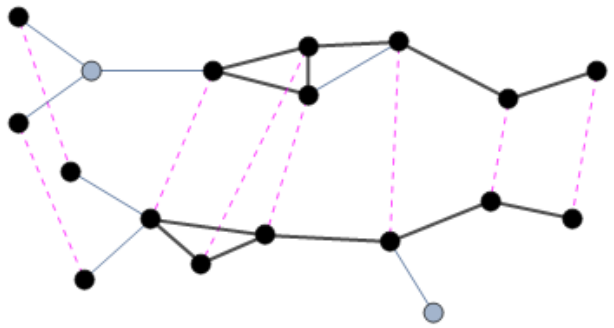


Figure 5.5: Global alignment of a network. Not all vertices must be mapped to a vertex in the other network, but the mapping must be one-to-one in both directions for those which are.

if it is locally suboptimal in some regions of the networks [75].

The local alignment strategies we observed are (mostly[7]) concerned with either searching for conserved motifs across multiple networks [54, 5, 28], or with identifying subnetworks in a given network which are similar to specific query subnetworks that are known to be functional [40, 71, 72, 28]. The PathBLAST algorithm introduced by Kelley et. al. in 2004 [40] and its successor NetworkBLAST [71] (which extends the algorithm to allow for comparison of multiple networks) both fall into the latter category. Neither performs alignments in an assignment problem style; PathBLAST allows both query pathways and query networks, but it handles query networks by searching over conserved pathways, while NetworkBLAST constructs an alignment graph for multiple species by matching vertices according to the sequence similarity of their corresponding proteins and defining fixed conditions under which an edge is considered to be conserved, rather than searching over the space of possible alignments.

Singh et. al. claimed in 2007 [75] that the global network alignment problem had previously received little attention in the literature. This claim is supported by our dataset, as no papers referencing "global alignment" or "global network alignment" (which address the topic as we have defined it) were published any earlier than 2007. Singh et al's paper appears to have been a turning point for global alignment, however; most of the papers in our dataset which address alignment address global alignment, and none of the local alignment papers in our reading list were published any later than 2006.

This transition seems to have occurred along with the goal of searching for increasingly large conserved subnetworks across species without necessarily knowing beforehand which regions are biologically significant, and with a shift towards using the topological information of networks to gain insight into vertex similarities, as an explosion of vast amounts of molecular interaction data became available. Full-scale comparison of networks with the goal of finding potentially *new* significant structural regions is a task for which global alignment is better suited than local, and so global alignment strategies became prevalent.

[7] [44] I'm not quite sure about yet

A ~~more~~ modern overview of the distinctions between and history of local and global network alignment was published by Guzzi and Milenković in 2017. [34].

5.5.2 Graemlin and $\mathbf{M_AWIS_H}$. These are the only other two local alignment ones and neither of them are particularly hungarian method.

[44] (MAWISH, still not super sure what it's doing; something something min-cut graph partitioning? idk)

Graemlin: Cited by IsoRank paper, definitely local, they said it's the "first LNA method to align multiple species simultaneously" [28] flannick2006graemlin

Pinalog refers to them: "MaWISH adopts the evolutionary models of match, mismatch, and deletion of the proteins", and "Graemlin infers an alignment from network modules".

VOMIT

5.6 Global alignment algorithms

??

We now present the alignment algorithms we found to use the assignment problem schemaf, including those in the field of pattern recognition as well as biology. This is not meant to be a comprehensive overview; we include all algorithms directly introduced in the high centrality vertices of our dataset, and have attempted to select the most important/influential of those which those papers discuss. We include the assignment problem-style pattern recognition algorithms from our reading list here, rather than in the pattern recognition chapter, in order to facilitate comparison of their methods and highlight the distinctions between the two fields.

In our presentation of each algorithm, we highlight:

- ~~What is~~ the similarity calculation strategy ~~here?~~
- What is the strategy used to search the similarity matrix? search strategy
- Why was this method introduced? What advantages does it have over earlier work?

Motivation for the method. • Advantages over other methods

Probably should define "maximum weight matching" somewhere but idk exactly where

Maybe in chapter 4

Name or Description	Year	Similarity Scoring	Alignment Construction
IsoRank [75]	2007	Convex combination of (eigenvalue problem-based) topological and external node similarities	Maximum-weight bipartite matching OR greedy pairing of highest scores
TODO Natalie [41]	2009	Convex combination of (general) node mapping scores and edge mapping scores	Cast as an integer linear programming problem and use Lagrangian relaxation
GRAAL [45]	2010	Convex combination of graphlet signatures and local density	Greedy neighborhood alignment around highest-scoring pairs (seed-and-extend)
PINALOG [61]	2012	External information (sequence and functional similarity of proteins), but includes topological similarity for extension mapping	Detect communities in each network, map similar proteins from communities onto each other, extend mapping to their neighbors
GHOST [59]	2012	Eigenvalue distributions of appropriately normalized neighborhood Laplacians	Seed-and-extend with approximate solutions to the quadratic assignment problem, then local search step
SPINAL [2]	2013	Convex combination of sequence and (neighbor matching-based) topological similarity	Seed-and-extend with local swaps
TODO NETAL [57]	2013	Topological (based on neighbors in a certain matching) + external	Greedy algorithm iterate on scores? something something
MAGNA [68]	2014	Any	Improve a population of existing alignments with crossover and a fitness function
Node fingerprinting [63]	2014	Minimize degree differences between neighbors while rewarding adjacency to already-matched node pairs (external info. optional)	Progressively add high-scoring pairings to an alignment and update scores accordingly
Node signatures [39]	2009	Vertex degree and incident edge weights	Hungarian method
Graph edit distance approximation [64]	2009	Edit costs (insertions, substitutions, deletions, plus their minimal ssociated edge edit costs)	Generalized (non-square) Munkres' algorithm
Modified GED approximation [69]	2014	Modification of edit costs when edit distance is a proper distance function	Generalized (non-square) Munkres' algorithm

Table 5.3: Broad summary of alignment algorithms discussed in this section. The distinctions between the various topological similarity scores used are discussed in each algorithm's individual section.

5.6.1 Non-biology methods. In our initial introduction of the assignment problem, we discussed how an approximation of the graph edit distance (GED) can be found by searching for an optimal matching within a matrix of costs[8] for specific edit operations, using a modified version of the Hungarian method [64]. An improved variant of Riesen and Bunke's 2009 algorithm was introduced by Serratosa in 2014, which uses the same modified Hungarian method, but defines a different and smaller matrix cost in the case where edit costs result in an edit distance which is an actual distance function (i.e. costs are nonnegative, substitution of identical-attribute nodes has zero cost, insertion and deletion have the same cost, and substitution costs no more than performing both an insertion and a deletion).

In the pattern recognition portion of our reading list, we saw one other assignment-problem style method, from Jouili and Tabbone in 2009 [39]. It uses the Manhattan distance on vectors with the degree of a node followed by the weights of its incident edges in decreasing order (the "node signatures") as a similarity score, and calculates the alignment itself simply by the Hungarian method. The method's results are reasonable but not overly impressive in relation to the (much older) methods the authors use for comparison; it appears to have been included in our dataset primarily because it cites other, more important papers, rather than because its own results are significant.

5.6.2 IsoRank. IsoRank was the first global network alignment method. Introduced by Singh et. al. in 2007 for the alignment of two networks [75], and extended the alignment of multiple networks in 2008 [74], it has remained a common benchmark for the performance of subsquent algorithms, with all other biological network alignment strategies we observed using it for comparison.

The authors calculate a similarity score between nodes by linearly interpolating between sequence similarity scores of proteins and topological similarity scores. The topological similarity score between vertex i in the source network V_1 and vertex j in the target network

[8] Most biology-specific alignment algorithms define a *similarity* score we would like to maximize, rather than a cost we would like to minimize, but we can obviously reformulate either strategy to match the other.

V_2 is defined to be the sum of the similarity scores for their neighbors, proportional to the total number of possible neighbor pairings. That is, we solve the eigenvalue problem

$$R_{ij} = \sum_{u \in N(i)} \sum_{v \in N(j)} \frac{R_{uv}}{|N(u)||N(v)|}, i \in V_1, j \in V_2,$$

where $N(u)$ is the number of neighbors of vertex u. The overall similarity score between vertices i and j is then the solution of the eigenvalue problem

$$R = \alpha AR + (1 - \alpha)E, \alpha \in [0, 1],$$

where E is a normalized vector of pairwise sequence similarity scores and A is a doubly indexed $|V_1||V_2| \times |V_1||V_2|$ matrix where $A[i, j][u, v] = 1/(N(u)N(v))$ if there is an edge from vertex i to u in V_1 and from vertex j to v in V_2, and zero otherwise; $A[i, j][u, v]$ refers to the entry at row (i, j) and column (u, v). The parameter α controls the weight of the topological data compared to the sequence data in the overall similarity scores.

This eigenvalue problem is solved via the power method, and Singh et. al. discuss two methods to construct an alignment; either constructing the maximum-weight bipartite matching, or a greedy method which repeatedly removes the highest-scoring pairs from consideration until the alignment is finished, and which the authors found to sometimes perform even better than the more principled algorithm. Once all nodes are aligned, the conserved edges are simply those which appear in both networks TODO phrase this sentence in a clear way my brain is fuzzy and it's not hard but I can't articulate it right

In 2014, Kollias et. al. introduced an adapted, parallelized version of IsoRank used to perform global alignment of networks two orders of magnitude larger than previously possible [42]. We consider this to fall under the umbrella of the IsoRank method, rather than being a new alignment algorithm in its own right.

5.6.3 Natalie. Introduced by Klau in 2009 [41], Natalie

5.6.4 GRAAL. Introduced by Kuchaiev et. al. in 2010 [45], GRAAL is unique in that it incorporates *only* topological information into its node similarity scores. These similarity scores are based on the **graphlet degree signature** of each node, which is simply a vector of the number of each type of graphlet that the node touches. As with the graphlet degree distribution, we distinguish between different automorphism orbits of each graphlet. Each of these orbits is assigned a weight w_i that accounts for dependencies between orbits[9], and the *signature similarity* between nodes $u \in G$ and $v \in H$ is then

$$D(u,v) = \frac{1}{\sum_{i=0}^{72} w_i} \left[w_i \times \frac{|\log(u_i+1) - \log(v_i+1)|}{\log(\max\{u_i, v_i\}+2)} \right],$$

where u_i is the number of times a node u is touched by orbit i in a graph G. The overall similarity between two nodes is then their signature similarity, scaled according to their relative degrees in order to favor aligning the densest parts of the networks first.

To construct an alignment using this similarity score matrix, GRAAL chooses an initial high-scoring pair, and then builds neighborhoods of all possible radii around each member of the pair. These are aligned using a greedy strategy. If this does not result in a match for all the vertices in the smaller network, the same strategy is repeated for the graph G^2 (whose edges run between nodes connected by a path of length up to 2 in G), G^3, and so on until all the nodes in the smaller network are aligned.

This has the advantage of being well suited for networks of very different size, in which a typical greedy alignment such as that used in IsoRank is not likely to produce a connected component in the larger graph. Most global alignment strategies published after GRAAL use some variation on a seed-and-extend-neighborhoods strategy like this one in order to favor connected components in the alignment result. The authors compare GRAAL to IsoRank and show greatly improved results for edge conservation and connected component size in the alignment of the PPI networks for yeast and fly.

[9]For example, differences in counts of orbit 3 will imply differences in counts of all orbits that contain a triangle, and it is therefore assigned a higher weight.

5.6.5 PINALOG. Introduced by Phan and Sternberg in 2012 [61], PINALOG computes a global alignment between protein interaction networks by detecting communities within each network by merging adjacent cliques, mapping highly similar proteins from these communities onto each other using the Hungarian method with vertex similarity scores based on sequence and functional similarity of their corresponding proteins, and finally extending the mapping to these highly similar proteins' neighbors to obtain the remainder of the alignment. The extension of the mapping to the neighbors of similar communities incorporates topological similarity as well as the external information of sequence and functional similarity scores in order to get a matrix of similarity scores between these neighbors in each network, from which optimal pairings are selected using the Hungarian method.

The authors compare their method to IsoRank, Graemlin 2.0 (an updated variant of the Graemlin algorithm previously described), and MI-GRAAL (an updated variant of the GRAAL algorithm previously described) with respect to various metrics and for various tasks. PINALOG was able to conserve more interactions than IsoRank, but fewer than MI-GRAAL, and shows a much higher conservation of interactions with functional similarity than either, which is expected given its incorporation of functional similarity scores into the algorithm. Overall results and claims of improvement over other methods were not fully summarized, with the authors instead discussing the difficulties in comparing performance of PPI network alignment methods.

5.6.6 GHOST. Introduced by Patro and Kingsford in 2012 [59], GHOST is a pairwise network alignment strategy which interpolates between topological and sequence distance information to get its overall node similarity scores. Its topological distance scores are based on the density functions of the spectra for the normalized Laplacian of various-radius neighborhoods of a given vertex. Density is used, rather than comparing the spectra themselves, as the length of each spectrum varies according to the size of the neighborhood it originates from. The distances between spectral densities for two vertices are averaged over several neighborhood radii to produce the final topological distance between them.

To align two networks using these scores, GHOST seeds regions of an alignment with high scoring pairs of nodes from the different networks and then extends the alignment around the neighborhoods of the two nodes, matching the neighborhoods by computing an approximate solution to the quadratic assignment problem[10]. This process continues until all nodes from the smaller network have been aligned to a node in the larger network, at which point regions of the solution space around the initial result are explored to potentially find a better solution.

The authors evaluate the performance of GHOST against IsoRank, GRAAL, MI-GRAAL, H-GRAAL (another variant in the GRAAL family), and Natalie 2.0 (an updated variant of the Natalie algorithm previously described; see [23]), and I REALLY CAN'T THINK STRAIGHT ANYMORE I WILL FIX THE ANALYSIS OF THEIR RESULTS LATER

fix

5.6.7 SPINAL. In SPINAL, introduced by Aladağ and Erten in 2013 [2], as in IsoRank and GHOST, similarity scores are a convex combination of a topological similarity score and sequence similarity score between a pair of proteins. Topological similarity scores are computed based on maximum potential conserved edges between neighbors of a potential matching pair, rather than simply being scaled by the product of their degrees as in IsoRank. The score matrix is calculated iteratively using a gradient-based method.

You keep using this term — did you define it anywhere?

The alignment is then constructed with a seed-and-extend method which grows connected components of the alignment around the highest-scoring unaligned pairs by constructing a maximum weight matching[11] for their neighbors and then checking if any improvements can be made via local swaps.

The authors then extensively compare SPINAL to IsoRank and MI-GRAAL, showing improved accuracy performance on the PPI networks for various organisms and noting SPINAL's reasonable running times compared to IsoRank and MI-GRAAL.

[10] Just like the assignment problem, except the cost function is expressed in terms of quadratic inequalities instead of being linear.

[11] i.e. a solution to the assignment problem. NO, THAT'S NOT QUITE RIGHT, FIX IT

5.6.8 NETAL. Interpolate between interaction scores and similarity scores.

Searching score matrix: Greedy strategy; choose the highest pairwise score, update interaction matrix to get the new matrix from which you're choosing the highest scores.

Just compared to GRAAL family and IsoRank itself

5.6.9 MAGNA. Introduced by Saraph and Milenkovič in 2014 [68], MAGNA is unique among the alignment strategies we observed in that it is not a specific alignment method, but rather a technique for improving upon existing alignments, which can be generated using any method. The key idea is the observation that existing alignment methods align similar nodes in hopes of maximizing the number of edges in a final alignment, since directly maximizing the number of conserved edges is intractable.

The authors use a genetic algorithm to improve upon a population of existing alignments, where the most "fit" alignments are those which maximizes the edge conservation in both the source and target networks being aligned. Their (novel) fitness function achieves this by penalizing both the mapping of sparse regions to dense regions and the mapping of dense regions to sparse regions, unlike previous alignment quality scoring methods. They also introduce a novel crossover method which takes the midpoint of the shortest path of transpositions between two parent alignments, which are sampled from a probability distribution of the fitness of all alignments in the population.

This method is shown by the authors to improve the results of initial populations of alignments generated randomly and by IsoRank, MI-GRAAL, and GHOST. Overall, it is able to outperform all these methods with respect to both node and edge conservation, and topological and biological alignment quality.

5.6.10 Node Fingerprinting. Node fingerprinting (NF), introduced by Radu and Charleston in 2014 [63], aims to quickly compute accurate alignments between two networks in a parallelizable manner without the need to rely on external information (such as protein sequence alignment similarity scores) or on tunable network alignment parameters which can intro-

duce an increased computational overhead. Their approach allows for the inclusion of such external information, but the authors choose to run experiments without it in order to avoid the circularity of using sequence information to both carry out and validate an alignment.

Like SPINAL and NETAL, the similarity scores used in node fingerprinting are updated throughout the process of constructing an alignment. These *pairing scores* are based on the relative differences in the in and out degrees (or simply the degree for an undirected network) of the neighbors of a potential pair to be matched; this difference is meant to be minimized. The scoring function adds a bonus for node pairings that are adjacent to already mapped node pairs, and a penalty for nodes with differing in or out degrees. The algorithm then repeatedly adds the node pairings with above average scores to the alignment and recalculates pairing scores until a complete alignment has been reached.

Like MAGNA, the authors compare their algorithm to IsoRank, MI-GRAAL, and GHOST. They run experiments on both real and synthetic data and show equal or improved accuracy, especially for large networks which contain more structural information than smaller ones. In some experiments (using smaller *Human Herpesvirus* networks), GHOST or MI-GRAAL is able to outperform NF, but at a greatly increased runtime and memory cost, while NF still returns reasonable results. The advantage of this method is thus primarily in the analysis of very large networks, where it is able to take advantage of increased structural information at a low overall computational cost.

Chapter 6. Conclusion

probably only really needs to be like a page or two

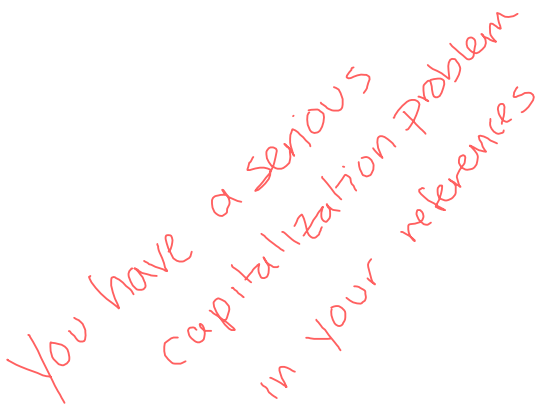
We have seen a lot of things. The connections–motifs and local stuff is actually kind of the most like pattern recognition. Tiny graphs, even if you're looking for them in a big graph. Search space pruning type strategies. Then the other place we see the connection is in assignment problem style stuff.

It was really super helpful throughout the process of writing to have all the cocitation

information handy, and to know why papers were included in the reading list. I didn't have to rely on my previous judgement about what's important, and if I was confused about why I had a paper on my reading list or wanted to include it, I just looked at who it cited and who cited it in the dataset. That's very helpful for a long writing process.

6.0.1 how CS stuff might be useful for bio, and barriers/limitations to that. they kind of already use the techniques and algorithms, they don't need it as much since it's not as specific. Ooh, we should see if the bio papers cite CS papers more than the CS papers cite bio. Edit distances is very analagous to string edit distances, which seems similar to sequence alignment.

6.0.2 how bio stuff might be useful for CS, etc.. BIG NETWORKS LIKE WITH LANGUAGE; CONSERVED SUBGRAPHS, ETC.? whole scenes, etc., external information, using graphlets and motif stuff for hashing purposes like in a large database



6.0.3 complain about the other surveys again and brag

on my way. Appendix A. Appendices

A.1 Additional Figures

Figure A.1: The sciMet network dataset used in our Table 2.1 comparison to G. Note the high number of connected components, and low number of children per parent, in contrast to the "blooming" behavior in G created by the inclusion of ALL child references from each parent paper.

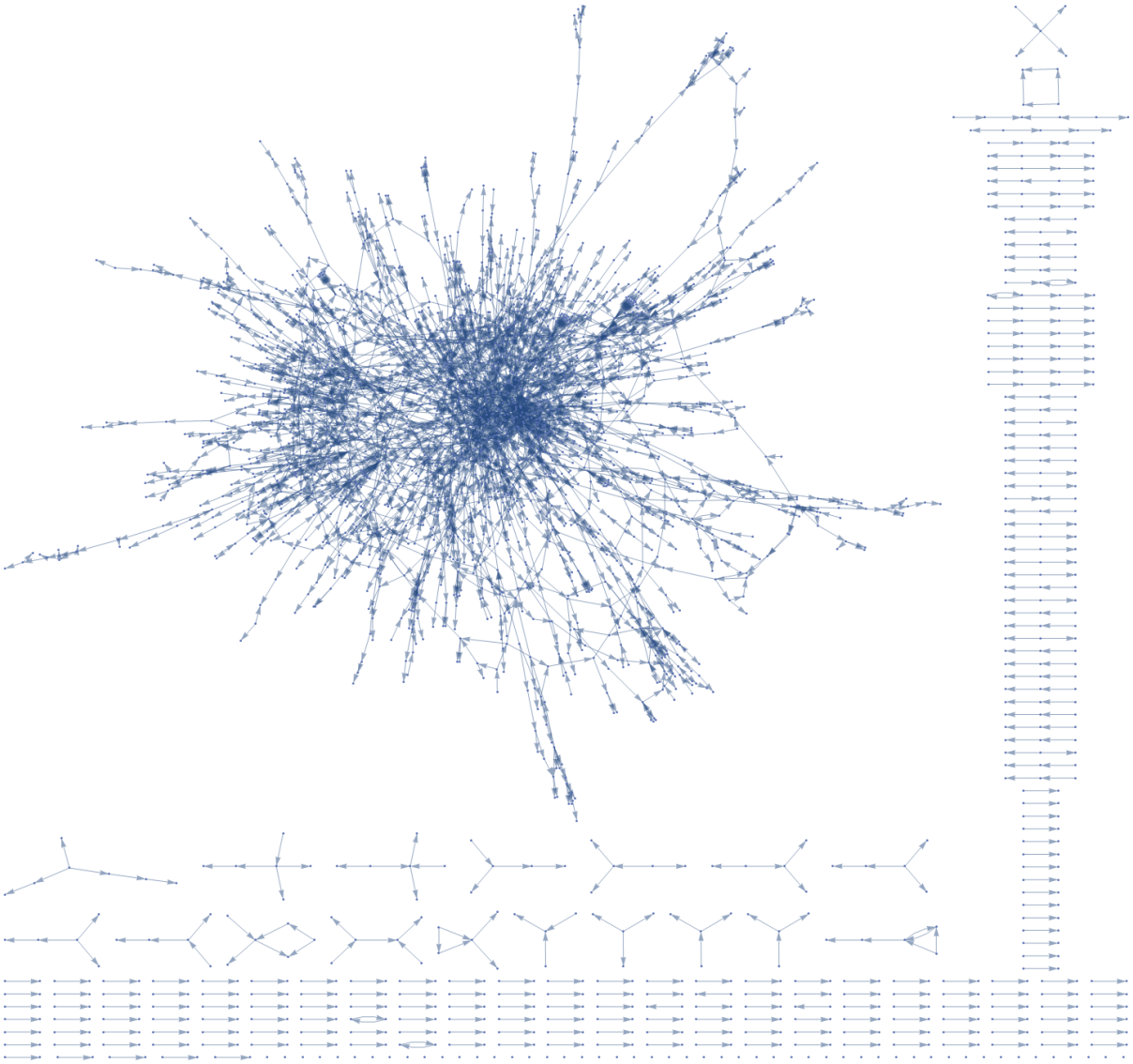


Figure A.2: The zewail citation network dataset used in our Table 2.1 comparison to G. Note the high number of connected components, and low number of children per parent, in contrast to the "blooming" behavior in G created by the inclusion of ALL child references from each parent paper.

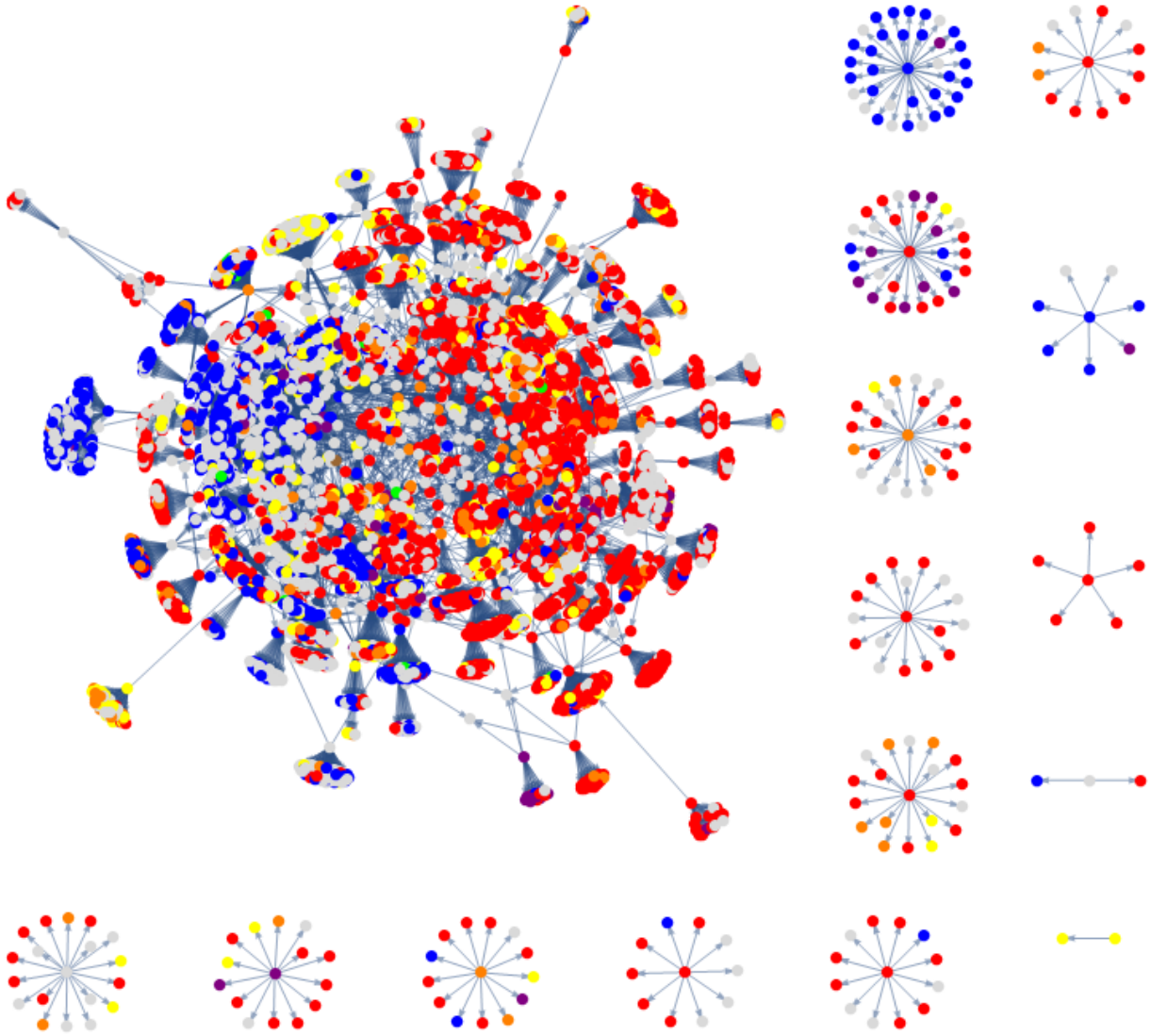


Figure A.3: The full network G_p, with vertices colored according to their subject label as in Figure 3.2.

A.2 Additional Tables

Subject	N	$G_R^{(1)}$	$G_R^{(2)}$	Title
None	22	15	7	Fifty years of graph matching, network alignment and network comparison
None	15	10	5	Networks for systems biology: conceptual connection of data and function
Biology	7	7	0	Global network alignment using multiscale spectral signatures
None	13	1	12	Error correcting graph matching: on the influence of the underlying cost function

Subject	N	$G_R^{(1)}$	$G_R^{(2)}$	Title
None	10	10	0	MAGNA: Maximizing Accuracy in Global Network Alignment
None	4	4	0	Graphlet-based measures are suitable for biological network comparison
None	7	6	1	On Graph Kernels: Hardness Results and Efficient Alternatives
Biology	8	8	0	Pairwise Alignment of Protein Interaction Networks
None	6	6	0	Alignment-free protein interaction network comparison
Biology	7	7	0	Biological network comparison using graphlet degree distribution
None	10	10	0	NETAL: a new graph-based method for global alignment of protein-protein interaction networks
CS/Math	10	4	6	Computers and Intractability: A Guide to the Theory of NP-Completeness
None	16	1	15	Recent developments in graph matching
None	10	10	0	Modeling cellular machinery through biological network comparison
CS	7	2	5	A graph distance metric based on the maximal common subgraph
None	24	1	23	Thirty years of graph matching in pattern recognition
None	6	6	0	Collective dynamics of “small-world” networks
None	13	11	2	A new graph-based method for pairwise global network alignment
None	12	4	8	An Algorithm for Subgraph Isomorphism
CS	8	2	6	On a relation between graph edit distance and maximum common subgraph
None	7	7	0	Topological network alignment uncovers biological function and phylogeny
CS	7	0	7	A linear programming approach for the weighted graph matching problem
None	10	0	10	An eigendecomposition approach to weighted graph matching problems
CS	8	0	8	A graduated assignment algorithm for graph matching
None	9	0	9	A new algorithm for subgraph optimal isomorphism
CS	9	0	9	A distance measure between attributed relational graphs for pattern recognition
CS	5	0	5	Inexact graph matching for structural pattern recognition
CS	6	2	4	A new algorithm for error-tolerant subgraph isomorphism detection
CS	6	0	6	Structural Descriptions and Inexact Matching
CS	6	0	6	A shape analysis model with applications to a character recognition system
CS	9	0	9	A graph distance measure for image analysis

Subject	N	$G_R^{(1)}$	$G_R^{(2)}$	Title
CS	8	0	8	Structural matching in computer vision using probabilistic relaxation
CS	6	0	6	Hierarchical attributed graph representation and recognition of handwritten chinese characters
CS	4	0	4	Linear time algorithm for isomorphism of planar graphs (Preliminary Report)
CS/Bio	6	5	1	Pairwise Global Alignment of Protein Interaction Networks by Matching Neighborhood Topology
None	6	6	0	Conserved patterns of protein interaction in multiple species
None	5	0	5	Approximate graph edit distance computation by means of bipartite graph matching
None	8	4	4	Local graph alignment and motif search in biological networks
None	6	6	0	Network Motifs: Simple Building Blocks of Complex Networks
None	8	1	7	The Hungarian method for the assignment problem
CS	9	0	9	Graph Matching Based on Node Signatures
None	7	0	7	Exact and approximate graph matching using random walks
None	6	6	0	Global alignment of multiple protein interaction networks with application to functional orthology detection
None	10	9	1	Fast parallel algorithms for graph similarity and matching
CS	9	0	9	Fast computation of Bipartite graph matching
CS	4	0	4	Fast and Scalable Approximate Spectral Matching for Higher Order Graph Matching
CS	4	0	4	A Probabilistic Approach to Spectral Graph Matching
CS	3	0	3	Graph matching applications in pattern recognition and image processing
None	10	1	9	The graph matching problem
Biology	4	4	0	Graph-based methods for analysing networks in cell biology
CS	10	4	6	BIG-ALIGN: Fast Bipartite Graph Alignment
None	7	0	7	A (sub)graph isomorphism algorithm for matching large graphs
Biology	4	4	0	Complex network measures of brain connectivity: Uses and interpretations
CS	7	0	7	Efficient Graph Similarity Search Over Large Graph Databases
3	4	3	1	Demadroid: Object Reference Graph-Based Malware Detection in Android
None	2	2	0	Indian sign language recognition using graph matching on 3D motion captured signs
None	6	5	1	Predicting Graph Categories from Structural Properties
None	10	10	0	Survey on the Graph Alignment Problem and a Benchmark of Suitable Algorithms

Subject	N	$G_R^{(1)}$	$G_R^{(2)}$	Title
CS/Math	12	0	12	Efficient Graph Matching Algorithms
CS	2	2	0	Early Estimation Model for 3D-Discrete Indian Sign Language Recognition Using Graph Matching
None	1	0	1	Unsupervised Domain Adaptation Using Regularized Hyper-Graph Matching

Table A.2: Number of vertices in the neighborhood of each paper in G_R, and how many vertices in it lie on each side of the partition.

A.3 Code-related

A.4 Subject Tagging Keywords

A.5 Reference List Guide

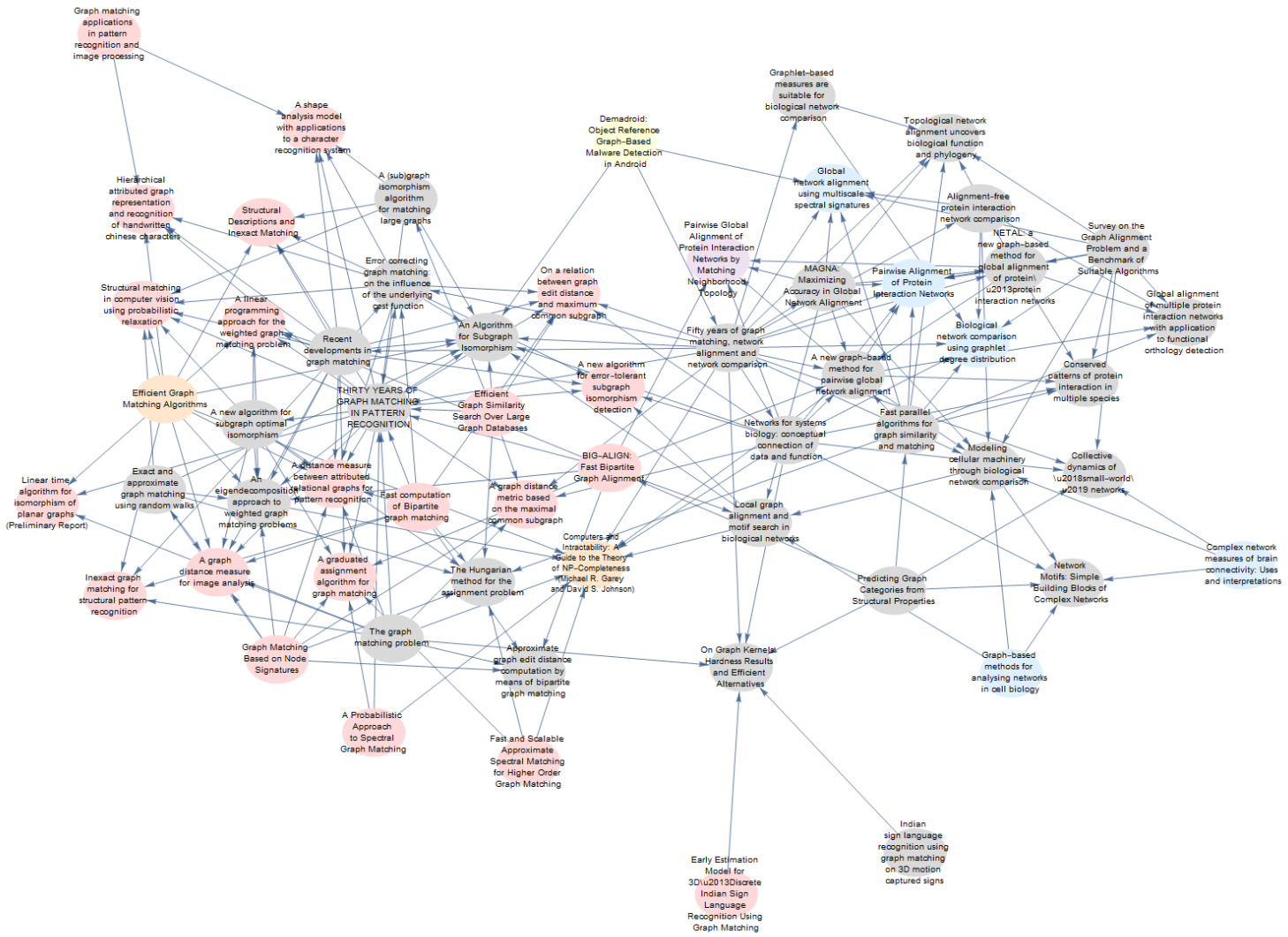


Figure A.4: The subnetwork S of high centrality papers, as listed in Tables 2.2, 2.3, and 2.4, with vertices colored according to their subject label. Grey vertices are unlabeled, pink is computer science, blue is biology, yellow is math, orange is both math and computer science, and purple is both computer science and biology.

Note: "Unsupervised Domain Adaptation Using Regularized Hyper-Graph Matching" is not in the connected component and is not displayed.

	G_R	$G_R^{(1)}$	$G_R^{(2)}$
Total vertices	61	27	34
Untagged	30	8	22
Tagged	31	19	12
CS	24	2	22
Biology	6	6	0
Math	3	1	2
Both CS and biology	1	1	0
Both CS and math	2	0	2
Both biology and math	0	0	0
All three	0	0	0

Table A.1: Number of vertices tagged as computer science, biology, math, or some combination of these in the reading list subnetwork G_R, and its intersections $G_R^{(1)}$ and $G_R^{(2)}$ with the two halves of the partition $G_p^{(1)}$ and $G_p^{(2)}$. See Table 3.1.

Usage	Package Name(s)
Mathematical Computation	NumPy NetworkX
Figure creation*	Matplotlib
File I/O Handling	csv glob re
API Request Handling	urllib Requests time
Interfacing with Google Sheets	gspread$^{\Diamond}$ oauth2client$^{\Diamond}$
The `defaultdict` datatype	collections
Interfacing my own modules with Jupyter notebooks	importlib$^{\Diamond}$

Table A.3: Python packages used for the project. All but those marked with a $\Diamond$ are found in either the standard library or available in Anaconda for Python 3.6 on 64-bit Windows in mid-2018, and the remainder can be installed via pip.

*Only Figure 1.3 was created in Python. The remainder were made with Mathematica.

Computer Science	Biology	Mathematics
ACM	Biochem-	Algebra
Algorithm	Biocomputing	Algorithm
Artificial Intelligence	Bioengineering	Chaos
CIVR	Bioinformatic	Combinatori-
Computational Intelligence	Biological	Fixed Point
Computational Linguistics	Biology	Fractal
Computer	Biomedic-	Functional Analysis
Computer Graphics	Biosystem	Geometr-
Computer Science	Biotechnology	Graph
Computer Vision	Brain	Kernel
Data	Cancer	Linear Regression
Data Mining	Cardiology	Markov
Document Analysis	Cell	Mathemati-
Electrical Engineering	Disease	Multivariate
Graphics	DNA	Network
IEEE	Drug	Optimization
Image Analysis	Endocrinology	Permutation Group
Image Processing	Epidemiology	Probability
Intelligent System	Genetic	Riemann Surface
Internet	Genome	SIAM
ITiCSE	Genomic	Statistic-
Language Processing	Medical	Topology
Learning	Medicinal	Wavelet
Machine Learning	Medicine	
Machine Vision	Metabolic	
Malware	Microbiology	
Neural Network	Molecular	
Pattern Recognition	Neuro-	
Robotic	Neurobiological	
Scientific Computing	Pathology	
SIAM	Pathogen	
Signal Processing	Pharma-	
Software	Plant	
World Wide Web	Protein	
	Proteom-	
	Psych-	
	Psychology	
	Virology	
	Virus	

Table A.4: Keywords used to tag journal names as various subjects.

*Note: Both a term and its plural are considered a match, and hyphens indicate a word with several ending variations which were all considered to be associated with the tag. While the search process was case sensitive in order to avoid false positives for short words like "ACM", case-insensitive duplicate words have been excluded from the table. The words "algorithm" and "SIAM" are considered to be both computer science and mathematics.

Table A.5: Guide to references in the bibliography

Bibliography

[1] T. Aittokallio. Graph-based methods for analysing networks in cell biology. *Briefings in Bioinformatics*, 7(3):243–255, may 2006.

[2] Ahmet E Aladağ and Cesim Erten. Spinal: scalable protein interaction network alignment. *Bioinformatics*, 29(7):917–924, 2013.

[3] Waqar Ali, Tiago Rito, Gesine Reinert, Fengzhu Sun, and Charlotte M. Deane. Alignment-free protein interaction network comparison. *Bioinformatics*, 30(17):i430–i437, aug 2014.

[4] H.A. Almohamad and S.O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):522–525, may 1993.

[5] J. Berg and M. Lassig. Local graph alignment and motif search in biological networks. *Proceedings of the National Academy of Sciences*, 101(41):14689–14694, sep 2004.

[6] François Bourgeois and Jean-Claude Lassalle. An extension of the munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14(12):802–804, 1971.

[7] H. Bunke. Recent developments in graph matching. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. IEEE Comput. Soc.

[8] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, aug 1997.

[9] H. Bunke. Error correcting graph matching: on the influence of the underlying cost function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):917–922, 1999.

[10] H Bunke and G Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4):245–253, may 1983.

[11] Horst Bunke and Kim Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3-4):255–259, mar 1998.

[12] James P Canning, Emma E Ingram, Sammantha Nowak-Wolff, Adriana M Ortiz, Nesreen K Ahmed, Ryan A Rossi, Karl RB Schmitt, and Sucheta Soundarajan. Predicting graph categories from structural properties. *arXiv preprint arXiv:1805.02682*, 2018.

[13] Vincenzo Carletti, Pasquale Foggia, Antonio Greco, Alessia Saggese, and Mario Vento. Comparing performance of graph matching algorithms on huge graphs. *Pattern Recognition Letters*, 2018.

[14] W.J. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):749–764, 1995.

[15] James R Clough, Daniel R Balfour, Paul K Marsden, Claudia Prieto, Andrew J Reader, and Andrew P King. Mri slice stacking using manifold alignment and wave kernel signatures. In *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*, pages 319–323. IEEE, 2018.

[16] D. Conte, P. Foggia, C. Sansone, and M. Vento. Graph matching applications in pattern recognition and image processing. In *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*. IEEE.

[17] D. CONTE, P. FOGGIA, C. SANSONE, and M. VENTO. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(03):265–298, may 2004.

[18] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1367–1372, oct 2004.

[19] Debasmit Das and CS Lee. Unsupervised domain adaptation using regularized hypergraph matching. *arXiv preprint arXiv:1805.08874*, 2018.

[20] Kexin Deng, Jie Tian, Jian Zheng, Xing Zhang, Xiaoqian Dai, and Min Xu. Retinal fundus image registration via vascular structure graph matching. *Journal of Biomedical Imaging*, 2010:14, 2010.

[21] Christoph Döpmann. Survey on the graph alignment problem and a benchmark of suitable algorithms. *Institut für Informatik*, 2013.

[22] Amir Egozi, Yosi Keller, and Hugo Guterman. A probabilistic approach to spectral graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):18–27, jan 2013.

[23] Mohammed El-Kebir, Jaap Heringa, and Gunnar W Klau. Lagrangian relaxation applied to sparse global network alignment. In *IAPR International Conference on Pattern Recognition in Bioinformatics*, pages 225–236. Springer, 2011.

[24] Yasser El-Sonbaty and M.A. Ismail. A new algorithm for subgraph optimal isomorphism. *Pattern Recognition*, 31(2):205–218, feb 1998.

[25] F. Emmert-Streib and M. Dehmer. Networks for systems biology: conceptual connection of data and function. *IET Systems Biology*, 5(3):185–207, may 2011.

[26] Frank Emmert-Streib, Matthias Dehmer, and Yongtang Shi. Fifty years of graph matching, network alignment and network comparison. *Information Sciences*, 346-347:180–197, jun 2016.

[27] M. A. Eshera and King-Sun Fu. A graph distance measure for image analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-14(3):398–408, may 1984.

[28] Jason Flannick, Antal Novak, Balaji S Srinivasan, Harley H McAdams, and Serafim Batzoglou. Graemlin: general and robust alignment of multiple large interaction networks. *Genome research*, 16(9):1169–1181, 2006.

[29] Pasquale Foggia, Gennaro Percannella, and Mario Vento. Graph matching and learning in pattern recognition in the last 10 years. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(01):1450001, 2014.

[30] François Fouss, Marco Saerens, and Masashi Shimbo. *Algorithms and models for network data and link analysis*. Cambridge University Press, 2016.

[31] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–388, apr 1996.

[32] M. Gori, M. Maggini, and L. Sarti. Exact and approximate graph matching using random walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1100–1111, jul 2005.

[33] Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*, pages 129–143. Springer Berlin Heidelberg, 2003.

[34] Pietro Hiram Guzzi and Tijana Milenković. Survey of local and global biological network alignment: the need to reconcile the two sides of the same coin. *Briefings in bioinformatics*, 19(3):472–481, 2017.

[35] Juris Hartmanis. Computers and intractability: A guide to the theory of np-completeness (michael r. garey and david s. johnson). *SIAM Review*, 24(1):90–91, jan 1982.

[36] W. Hayes, K. Sun, and N. Przulj. Graphlet-based measures are suitable for biological network comparison. *Bioinformatics*, 29(4):483–491, jan 2013.

[37] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *Proceedings of the sixth annual ACM symposium on Theory of computing - STOC '74*. ACM Press, 1974.

[38] DK Isenor and Safwat G Zaky. Fingerprint identification using graph matching. *Pattern Recognition*, 19(2):113–122, 1986.

[39] Salim Jouili and Salvatore Tabbone. Graph matching based on node signatures. In *Graph-Based Representations in Pattern Recognition*, pages 154–163. Springer Berlin Heidelberg, 2009.

[40] Brian P Kelley, Bingbing Yuan, Fran Lewitter, Roded Sharan, Brent R Stockwell, and Trey Ideker. Pathblast: a tool for alignment of protein interaction networks. *Nucleic acids research*, 32(suppl_2):W83–W88, 2004.

[41] Gunnar W Klau. A new graph-based method for pairwise global network alignment. *BMC Bioinformatics*, 10(Suppl 1):S59, 2009.

[42] Giorgos Kollias, Madan Sathe, Olaf Schenk, and Ananth Grama. Fast parallel algorithms for graph similarity and matching. *Journal of Parallel and Distributed Computing*, 74(5):2400–2410, may 2014.

[43] Danai Koutra, Hanghang Tong, and David Lubensky. BIG-ALIGN: Fast bipartite graph alignment. In *2013 IEEE 13th International Conference on Data Mining*. IEEE, dec 2013.

[44] Mehmet Koyutürk, Yohan Kim, Umut Topkara, Shankar Subramaniam, Wojciech Szpankowski, and Ananth Grama. Pairwise alignment of protein interaction networks. *Journal of Computational Biology*, 13(2):182–199, mar 2006.

[45] O. Kuchaiev, T. Milenkovic, V. Memisevic, W. Hayes, and N. Przulj. Topological network alignment uncovers biological function and phylogeny. *Journal of The Royal Society Interface*, 7(50):1341–1354, mar 2010.

[46] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, mar 1955.

[47] D. Anil Kumar, A. S. C. S. Sastry, P. V. V. Kishore, and E. Kiran Kumar. Indian sign language recognition using graph matching on 3d motion captured signs. *Multimedia Tools and Applications*, jun 2018.

[48] E Kiran Kumar, PVV Kishore, D Anil Kumar, and M Teja Kiran Kumar. Early estimation model for 3d–discrete indian sign language recognition using graph matching. *Journal of King Saud University-Computer and Information Sciences*, 2018.

[49] Thanh Nam Le, Muhammad Muzzamil Luqman, Anjan Dutta, Pierre Héroux, Christophe Rigaud, Clément Guérin, Pasquale Foggia, Jean-Christophe Burie, Jean-Marc Ogier, Josep Lladós, et al. Ssgci: Subgraph spotting in graph representations of comic book images. *Pattern Recognition Letters*, 2018.

[50] Lorenzo Livi and Antonello Rizzi. The graph matching problem. *Pattern Analysis and Applications*, 16(3):253–283, aug 2012.

[51] Si Wei Lu, Ying Ren, and Ching Y. Suen. Hierarchical attributed graph representation and recognition of handwritten chinese characters. *Pattern Recognition*, 24(7):617–632, jan 1991.

[52] Bruno T Messmer. *Efficient graph matching algorithms*. PhD thesis, 1995.

[53] B.T. Messmer and H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):493–504, may 1998.

[54] R. Milo. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, oct 2002.

[55] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[56] Mark Newman. *Networks: An Introduction*. OUP Oxford, 2010.

[57] Behnam Neyshabur, Ahmadreza Khadem, Somaye Hashemifar, and Seyed Shahriar Arab. Netal: a new graph-based method for global alignment of protein–protein interaction networks. *Bioinformatics*, 29(13):1654–1662, 2013.

[58] Soonyong Park, Sung-Kee Park, and Martial Hebert. Fast and scalable approximate spectral matching for higher order graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):479–492, mar 2014.

[59] R. Patro and C. Kingsford. Global network alignment using multiscale spectral signatures. *Bioinformatics*, 28(23):3105–3114, oct 2012.

[60] pepan (StackOverflow contributer). `https://stackoverflow.com/questions/17480142/is-there-any-simple-example-to-explain-ullmann-algorithm`, 2013. [Online; accessed 30-August-2018].

[61] Hang TT Phan and Michael JE Sternberg. Pinalog: a novel approach to align protein interaction networks—implications for complex detection and function prediction. *Bioinformatics*, 28(9):1239–1245, 2012.

[62] N. Przulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, jan 2007.

[63] Alex Radu and Michael Charleston. Node fingerprinting: an efficient heuristic for aligning biological networks. *Journal of Computational Biology*, 21(10):760–770, 2014.

[64] Kaspar Riesen and Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(7):950–959, jun 2009.

[65] J. Rocha and T. Pavlidis. A shape analysis model with applications to a character recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(4):393–404, apr 1994.

[66] Mikail Rubinov and Olaf Sporns. Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, 52(3):1059–1069, sep 2010.

[67] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(3):353–362, may 1983.

[68] Vikram Saraph and Tijana Milenković. Magna: Maximizing accuracy in global network alignment. *Bioinformatics*, 30(20):2931–2940, jul 2014.

[69] Francesc Serratosa. Fast computation of bipartite graph matching. *Pattern Recognition Letters*, 45:244–250, aug 2014.

[70] Linda G. Shapiro and Robert M. Haralick. Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(5):504–519, sep 1981.

[71] 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. *Proceedings of the National Academy of Sciences*, 102(6):1974–1979, feb 2005.

[72] Roded Sharan and Trey Ideker. Modeling cellular machinery through biological network comparison. *Nature Biotechnology*, 24(4):427–433, apr 2006.

[73] Harshita Sharma, Alexander Alekseychuk, Peter Leskovsky, Olaf Hellwich, RS Anand, Norman Zerbe, and Peter Hufnagl. Determining similarity in histological images using graph-theoretic description and matching methods for content-based image retrieval in medical diagnostics. *Diagnostic pathology*, 7(1):134, 2012.

[74] R. Singh, J. Xu, and B. Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences*, 105(35):12763–12768, aug 2008.

[75] Rohit Singh, Jinbo Xu, and Bonnie Berger. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In *Lecture Notes in Computer Science*, pages 16–31. Springer Berlin Heidelberg.

[76] J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, jan 1976.

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

[78] Huanran Wang, Hui He, and Weizhe Zhang. Demadroid: Object reference graph-based malware detection in android. *Security and Communication Networks*, 2018:1–16, may 2018.

[79] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440, 1998.

[80] Wikipedia contributors. Maximum common induced subgraph — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Maximum-common-induced-subgraph{&}oldid=749430208`, 2016. [Online; accessed 30-August-2018].

[81] Wikipedia contributors. Glossary of graph theory terms — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Glossary-of-graph-theory-terms{&}oldid=856668557`, 2018. [Online; accessed 29-August-2018].

[82] Wikipedia contributors. Graph isomorphism problem — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Graph-isomorphism-problem{&}oldid=854631915, 2018. [Online; accessed 30-August-2018].

[83] Wikipedia contributors. Induced subgraph — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Induced-subgraph{&}oldid=851218866, 2018. [Online; accessed 29-August-2018].

[84] Wikipedia contributors. Matching (graph theory) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Matching-(graph_theory){&}oldid=854762494, 2018. [Online; accessed 30-August-2018].

[85] Wikipedia contributors. Np-completeness — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=NP-completeness{&}oldid=841292328, 2018. [Online; accessed 29-August-2018].

[86] Wikipedia contributors. Np (complexity) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=NP_(complexity){&}oldid=856253902, 2018. [Online; accessed 29-August-2018].

[87] Wikipedia contributors. Np-hardness — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=NP-hardness{&}oldid=856021335, 2018. [Online; accessed 29-August-2018].

[88] Wikipedia contributors. Subgraph isomorphism problem — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Subgraph-isomorphism-problem{&}oldid=835246151, 2018. [Online; accessed 30-August-2018].

[89] Weiguo Zheng, Lei Zou, Xiang Lian, Dong Wang, and Dongyan Zhao. Efficient graph similarity search over large graph databases. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):964–978, apr 2015.

Index