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 goals and methods of graph and network comparison in each, with the ~~goal~~ 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 ~~goals~~ 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 six 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

Potential glossary terms (for glossary terms, note if Wikipedia has a good def'n or not)

- ortholog/homolog/paralog
- univariate measure/bivariate measure
- network motifs/graphlets
- attributed relational graph
- bipartite graph?
- NP hard, complete
- elastic graph matching (because it's NOT ACTUALLY GRAPH MATCHING)
- node neighborhood?
- weight matching/hungarian algorithm
- deterministic vs random network?
- PageRank?
- interactome
-

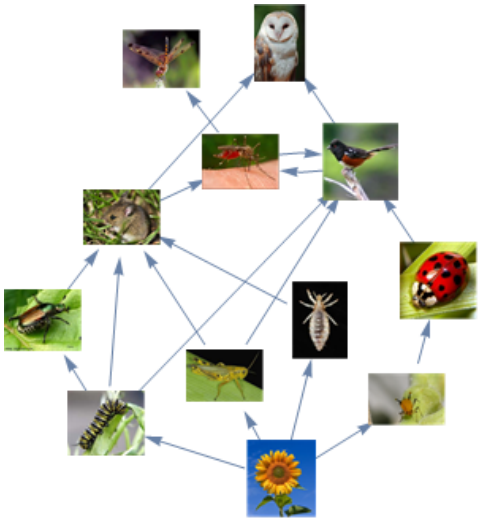
Questions/Notes

- There's a list in the one appendix about things I think I should probably include–mostly figures
- Commas in four digit numbers?
- Where should the reading list network pictures go? Same place as tables, in chapter 3, or in an appendix?

Notes for me

- Avoid contractions
- Fix figure references by labeling at the BOTTOM of the figure block, has to come after the caption
- Say which python packages are used, and include a reference for non-standard ones
- No more than one footnote every 2-3 pages.
- No rhetorical questions outside the introduction.
- Yes, cite CrossRef for sure.

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?

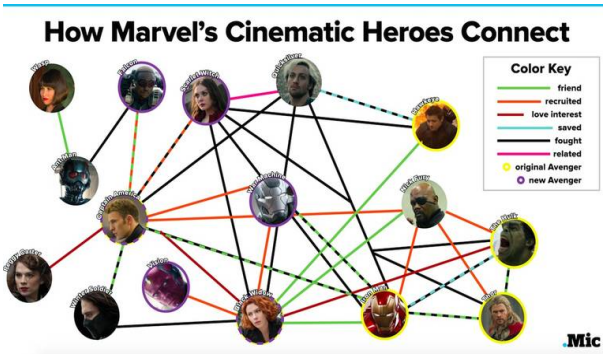


Diagram from the mic.com article "Here's How the Marvel Cinematic Universe's Heroes Connect–in One Surprising Map".

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?"[19]

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[6] 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 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 certain 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.

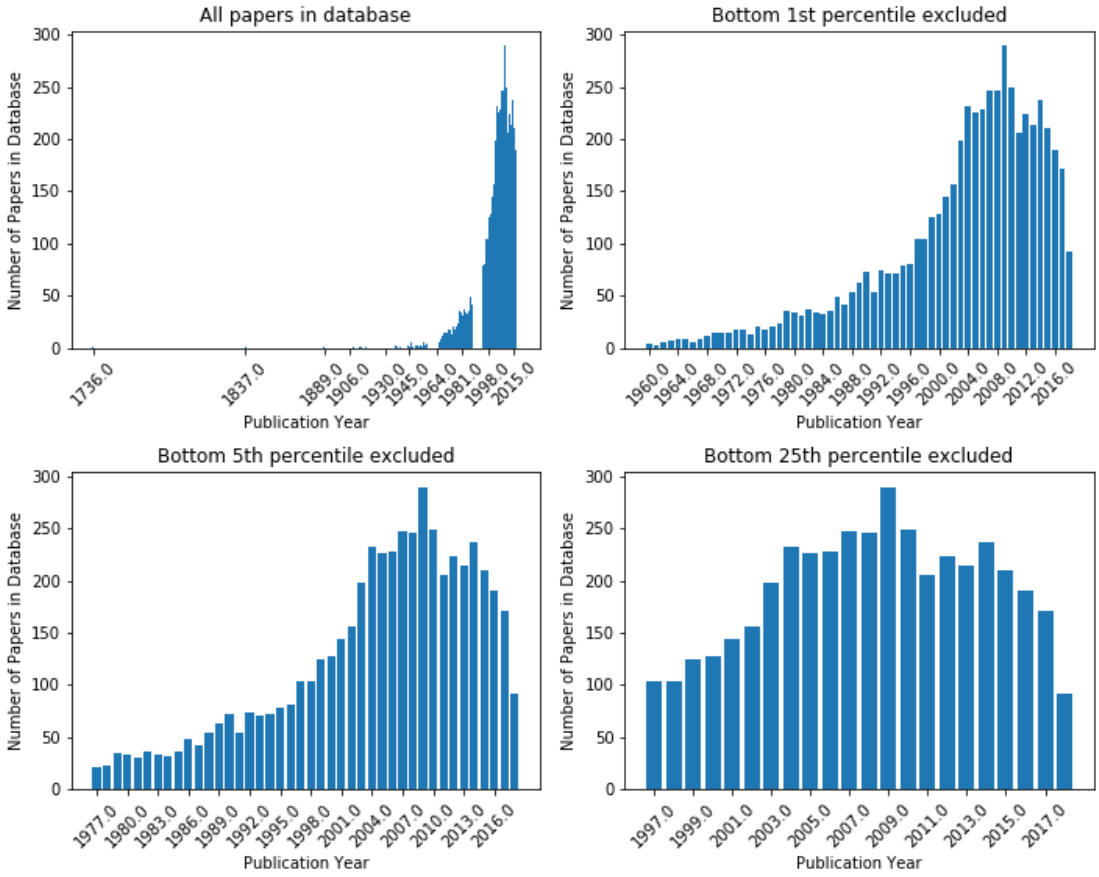


Figure 1.3: Distribution of papers published by year in our citation network of network similarity-related papers. Interestingly, year cutoffs for significant percentiles seem to roughly correspond to the spread of computers, personal computers, and the Internet, respectively.

1.2 PUT SOMETHING IN THE INTRODUCTION ABOUT THE TWO FIELDS AND STUFF

So, in our investigation, we ended up with two main fields, and we'll justify that. Here's a brief mention/summary that yay, our method is a nice way to do it.

Now, here's how the rest of the paper is going to be structured: In chapter 2, we talk about how we made our dataset, and introduce the tables of high centrality vertices that will become our reading list. In chapter 3, we talk about how we partitioned the dataset and discovered/justified the two communities thing, and brag on our approach and how nice it is to read this way. In chapter 4, we introduce the motivation and problem formulations for graph matching in pattern recognition, define and overview, etc., we will get to these

sentences later after the chapter is written.

1.3 Basic Network Properties

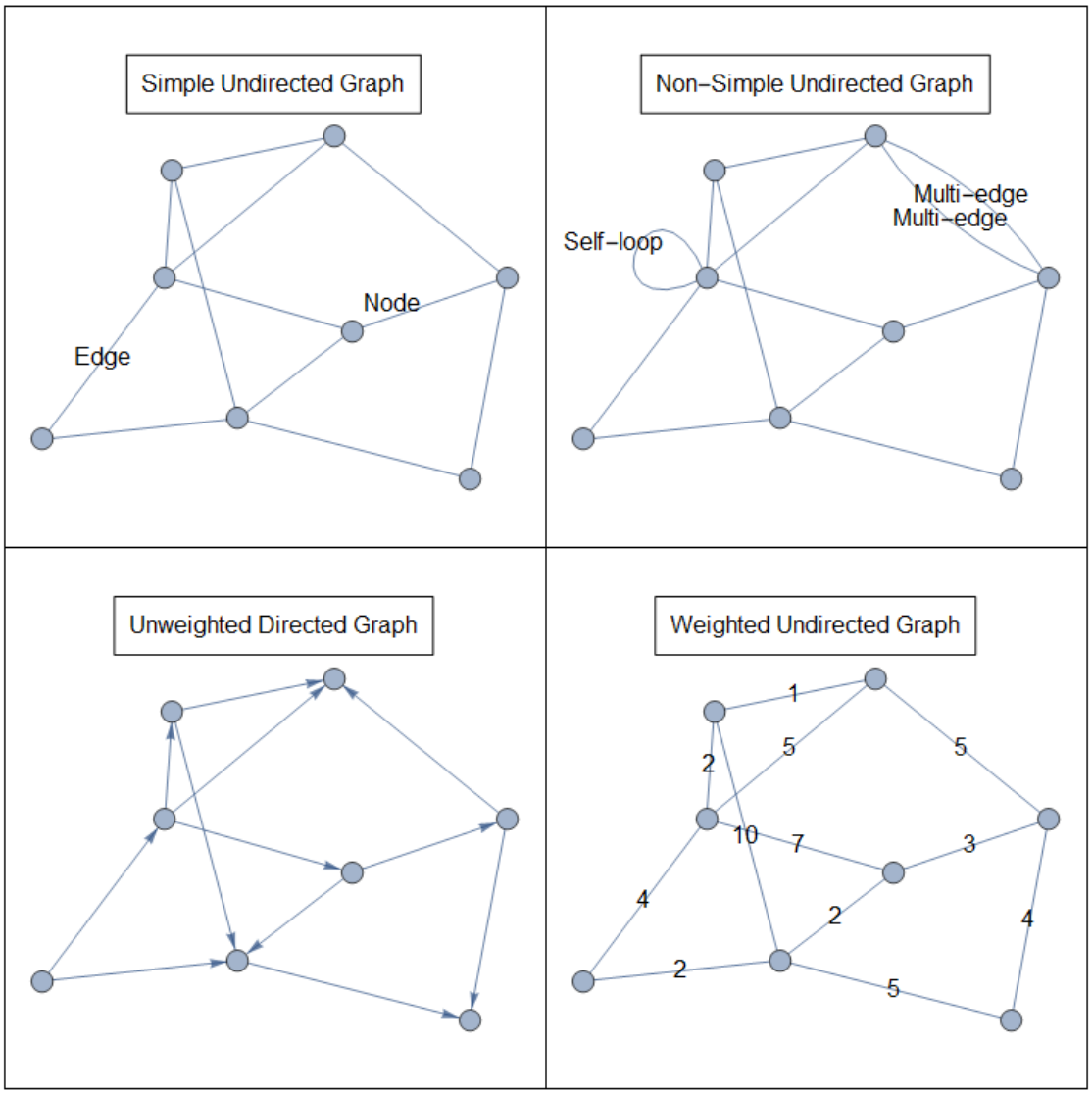


Figure 1.4: Basic types of networks

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*[17] closely, with the remainder of definitions not otherwise cited sourced from *Algorithms and Models for Network Data and Link Analysis*[9]

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

1.4 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.

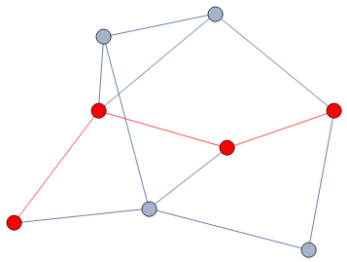


Figure 1.5: A path of length three on a small network.

1.4.1 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.

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.

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

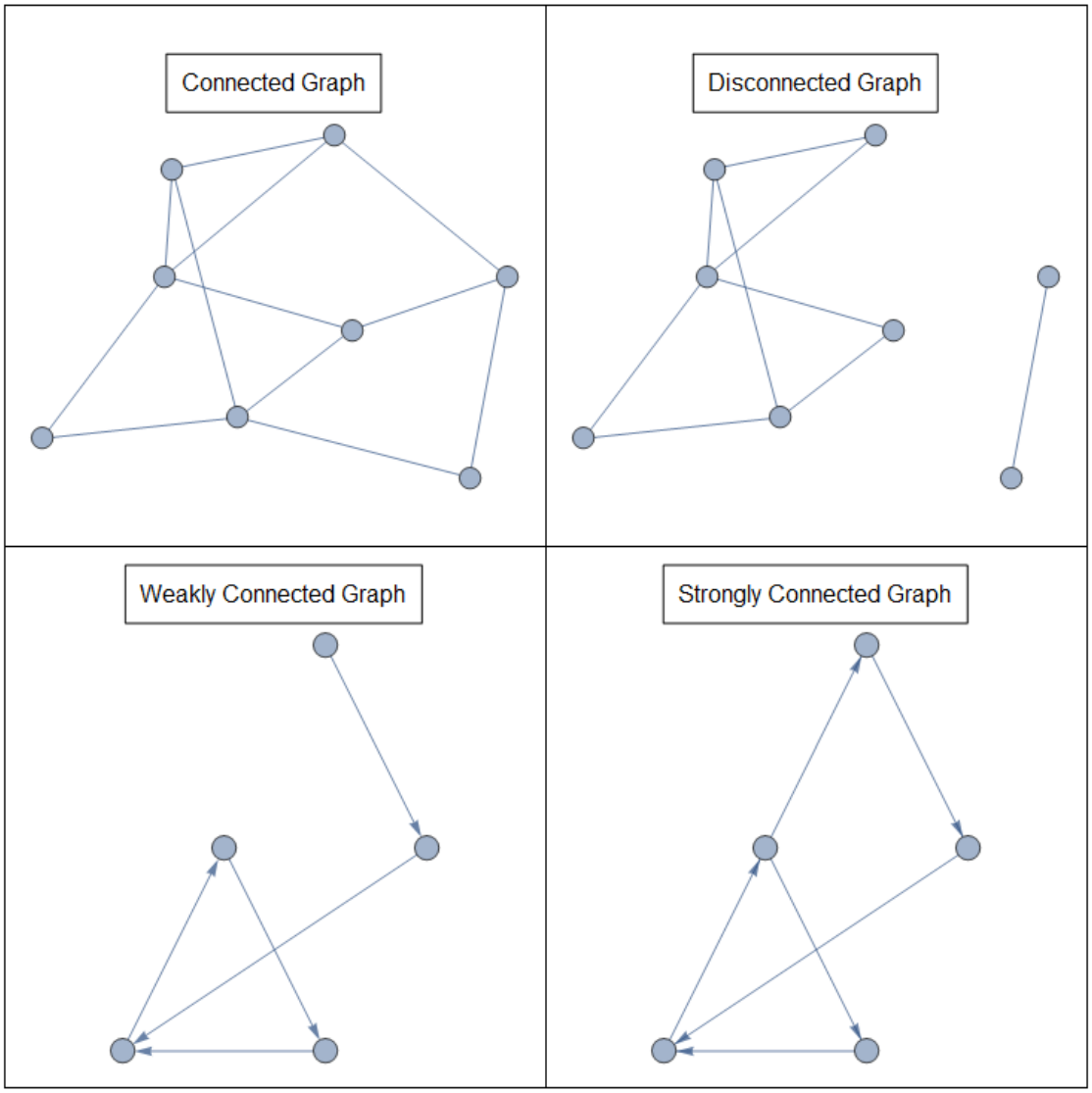
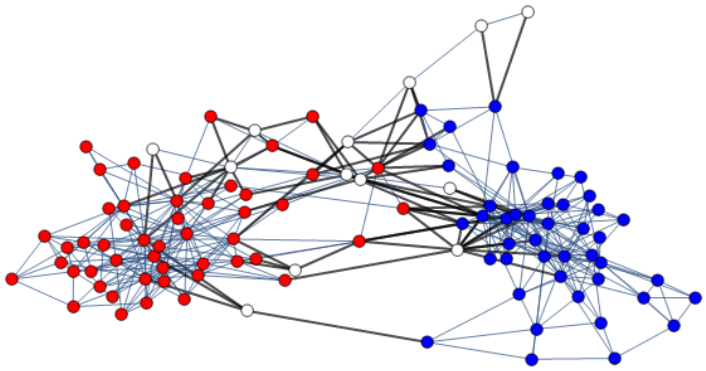


Figure 1.6: Simple examples of different types of network connectivity.

being connected; the higher the percentage of vertices are in the giant component, the closer the network is to being connected.

1.4.2 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.



"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.

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.

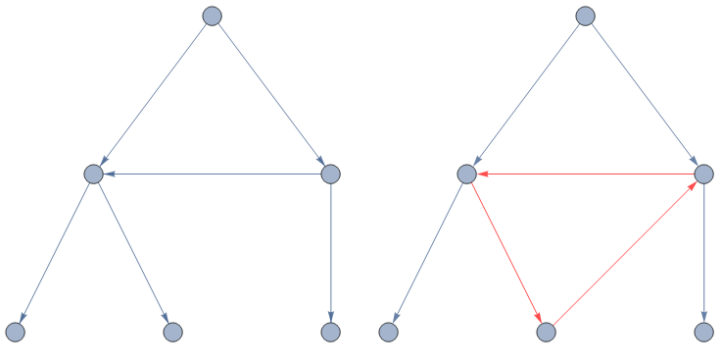


Figure 1.8: An acyclic directed network (left) vs. one which contains a cycle (right).

1.4.3 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. Dataset Creation and Analysis

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". 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 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[1], 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

[1]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.

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.

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 ?? 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 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

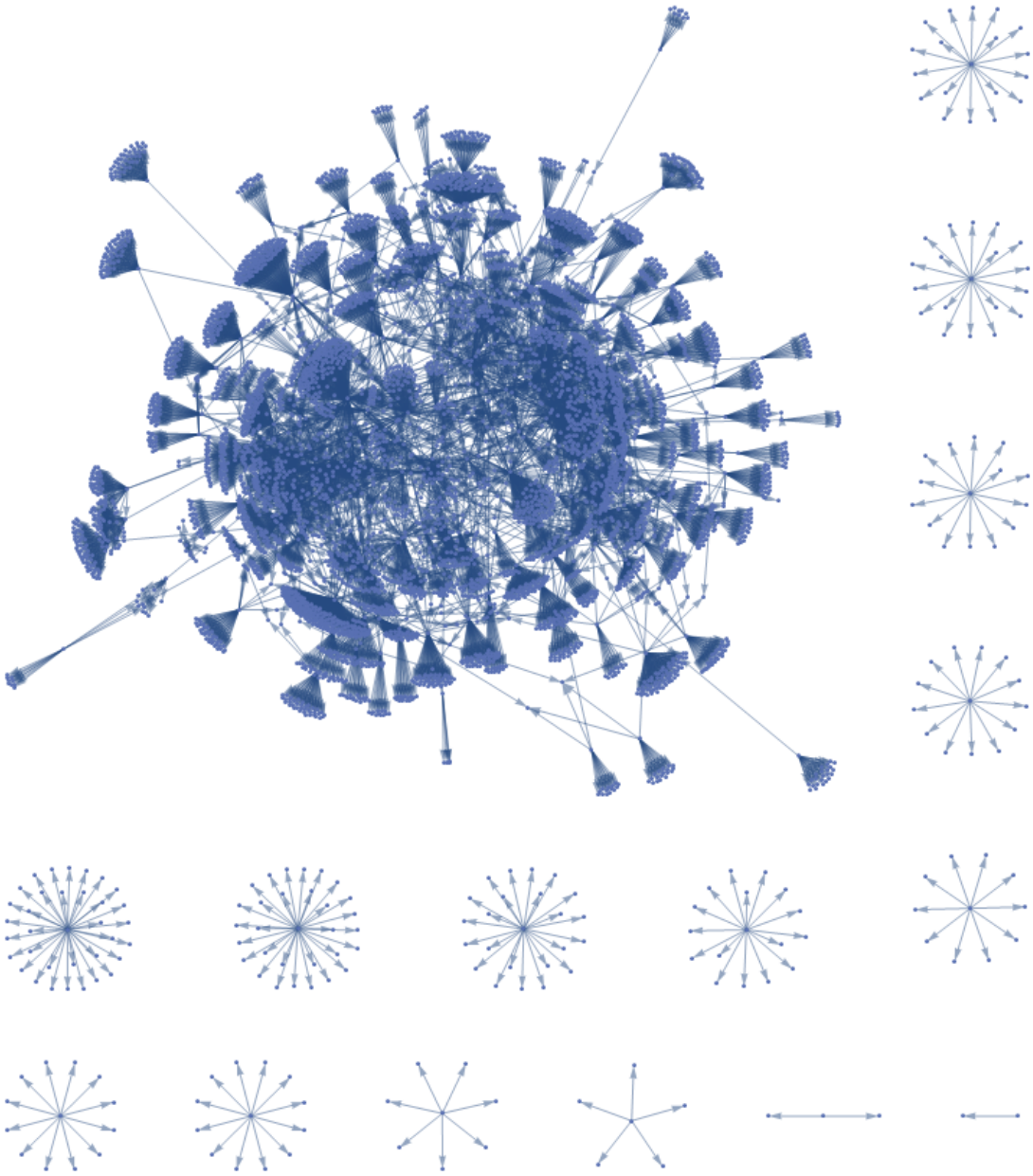


Figure 2.1: The full citation network of the dataset used for the project.

Figure 2.2: The pruned citation network.

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.

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 dis-

connected 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 only shrinks with a higher number of disconnected parent vertices, as fewer single-parent vertices become available compared to the rest.

2.3 Centrality Measures

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[2], 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

[2]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.

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.

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.

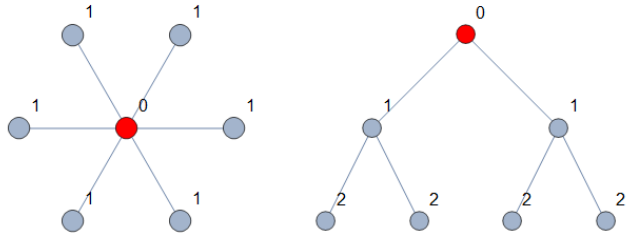


Figure 2.3: Two graphs with vertices labeled by their geodesic distance from the highlighted vertex.

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

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

2.4 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.[3] 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.

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

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

†Also top for Group 1 (biology dominated); *Also top for Group 2 (computer science dominated)

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	2	90	1	2		1
*Fifty years of graph matching, network alignment and network comparison	4	56	2	1		2
*Modeling cellular machinery through biological network comparison	9	40	4	3	10	9
*MAGNA: Maximizing Accuracy in Global Network Alignment	5	35	7	6		3
*On Graph Kernels: Hardness Results and Efficient Alternatives	10	9	3	8		
Biological network comparison using graphlet degree distribution	11	0		7	4	7
*A new graph-based method for pairwise global network alignment	8	12	9	4	6	
Network Motifs: Simple Building Blocks of Complex Networks	11	0		9	8	
*Pairwise Global Alignment of Protein Interaction Networks by Matching Neighborhood Topology	12	0			3	
*Topological network alignment uncovers biological function and phylogeny	12	0			2	
NETAL: a new graph-based method for global alignment of protein-protein interaction networks	6	26				5
Collective dynamics of “small-world” networks	10	0		10	5	
Global network alignment using multiscale spectral signatures	11	0			9	
*Global alignment of multiple protein interaction networks with application to functional orthology detection	10	0				
Conserved patterns of protein interaction in multiple species	10	0			7	
Pairwise Alignment of Protein Interaction Networks	10	0			1	
Alignment-free protein interaction network comparison	2	22	6	5		
Graphlet-based measures are suitable for biological network comparison	1	30				8
Survey on the Graph Alignment Problem and a Benchmark of Suitable Algorithms	0	26				4
Predicting Graph Categories from Structural Properties	0	30	5			
Fast parallel algorithms for graph similarity and matching	1	23				6
Complex network measures of brain connectivity: Uses and interpretations	0	28	8			
Graph-based methods for analysing networks in cell biology	0	30				10
Demadroid: Object Reference Graph-Based Malware Detection in Android	0	25	10			
Early Estimation Model for 3D-Discrete Indian Sign Language Recognition Using Graph Matching	0	29				
Indian sign language recognition using graph matching on 3D motion captured signs	0	29				

*Also a top-centrality paper for the entire network.

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	107	1	1		1
*An Algorithm for Subgraph Isomorphism	15	2	10	5	2	
*A graduated assignment algorithm for graph matching	18	0	7	4	3	
*An eigendecomposition approach to weighted graph matching problems	15	5		2	4	
*The graph matching problem	2	36	3	3		8
*A distance measure between attributed relational graphs for pattern recognition	13	0		7	1	
*Recent developments in graph matching	0	50	8			2
*Error correcting graph matching: on the influence of the underlying cost function	9	16		8		6
*Fast and Scalable Approximate Spectral Matching for Higher Order Graph Matching	0	41	2			
*Efficient Graph Matching Algorithms	0	42	5			4
*Computers and Intractability: A Guide to the Theory of NP-Completeness (Michael R. Garey and David S. Johnson)	11	0	6			
*The Hungarian method for the assignment problem	14	0				
*Graph matching applications in pattern recognition and image processing	0	40				3
Efficient Graph Similarity Search Over Large Graph Databases	0	28	4	6		
A linear programming approach for the weighted graph matching problem	8	8		9	9	
*Structural matching in computer vision using probabilistic relaxation	9	0			5	
*A graph distance measure for image analysis	8	0			6	
Inexact graph matching for structural pattern recognition	10	0				
*A new algorithm for subgraph optimal isomorphism	2	21				5
Approximate graph edit distance computation by means of bipartite graph matching	9	0				
Linear time algorithm for isomorphism of planar graphs (Preliminary Report)	9	0				
Structural Descriptions and Inexact Matching	9	0			7	
*A (sub)graph isomorphism algorithm for matching large graphs	3	16				7
A Probabilistic Approach to Spectral Graph Matching	0	25	9	10		
Hierarchical attributed graph representation and recognition of handwritten chinese characters	6	0			8	
Exact and approximate graph matching using random walks	1	14				9
A shape analysis model with applications to a character recognition system	5	0			10	
Fast computation of Bipartite graph matching	1	23				
Graph Matching Based on Node Signatures	0	17				10
Unsupervised Domain Adaptation Using Regularized Hyper-Graph Matching	0	22				

*Also a top-centrality paper for the entire network.

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

Chapter 3. I still don't know what to call this one tbh

3.1 Partitioning and Tagging the Dataset

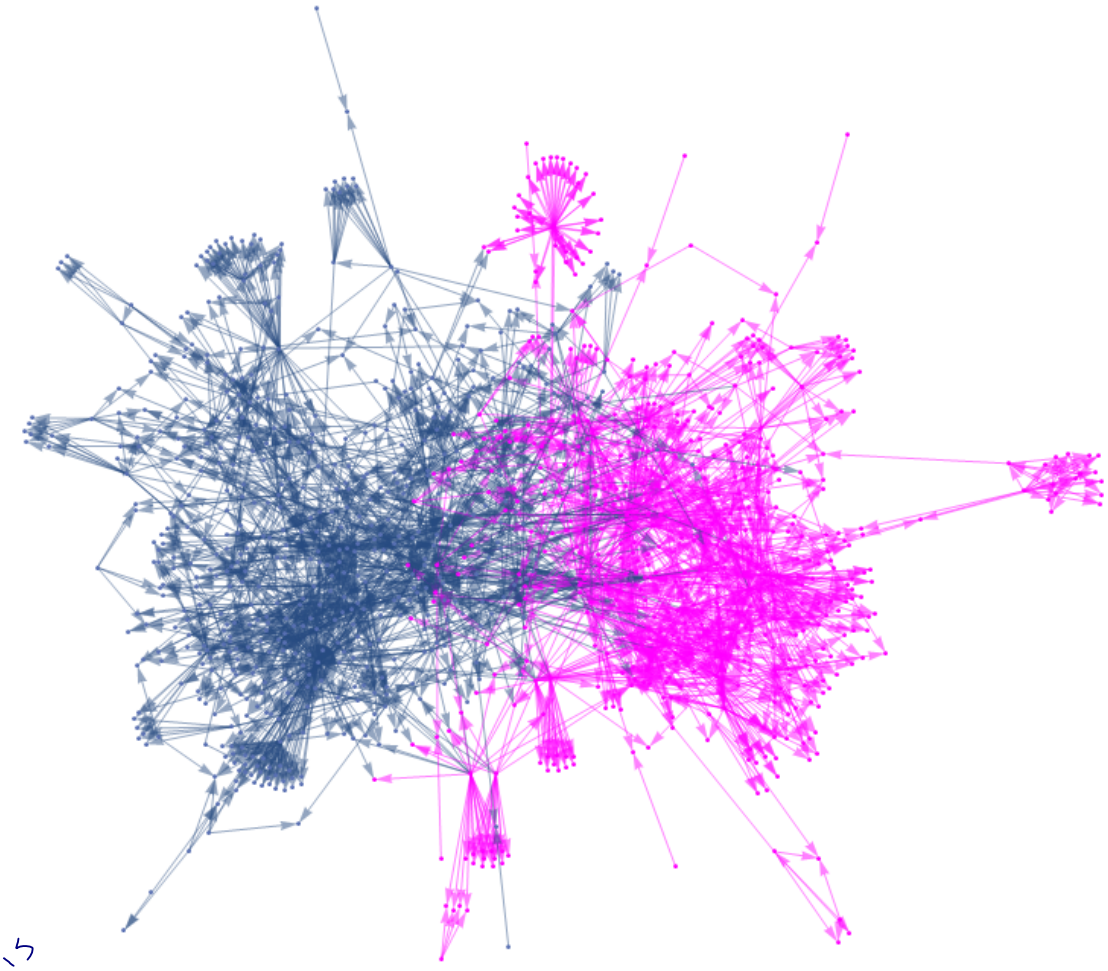


Figure 3.1: The two halves of our partition of the pruned network.

MAYBE DON'T SAY THIS

In the process of collecting relevant papers for our citation network, we read the abstracts for all of the parent papers, and over half the titles of their child references as we manually checked and parsed the freeform citations the CrossRef API was unable to automatically verify. While doing so, 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 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 Appendix ?. 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. This is far from perfect, but it is enough to confirm our initial suspicions. Our goal in the remainder this chapter is to investigate how well our categories are reflected in the structure of the dataset, both overall and with respect to our partition, and then

[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.

[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.

to broadly compare and contrast the study of network similarity in the fields of computer science and biology as an introduction to our discussion of each one specifically.

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

[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.

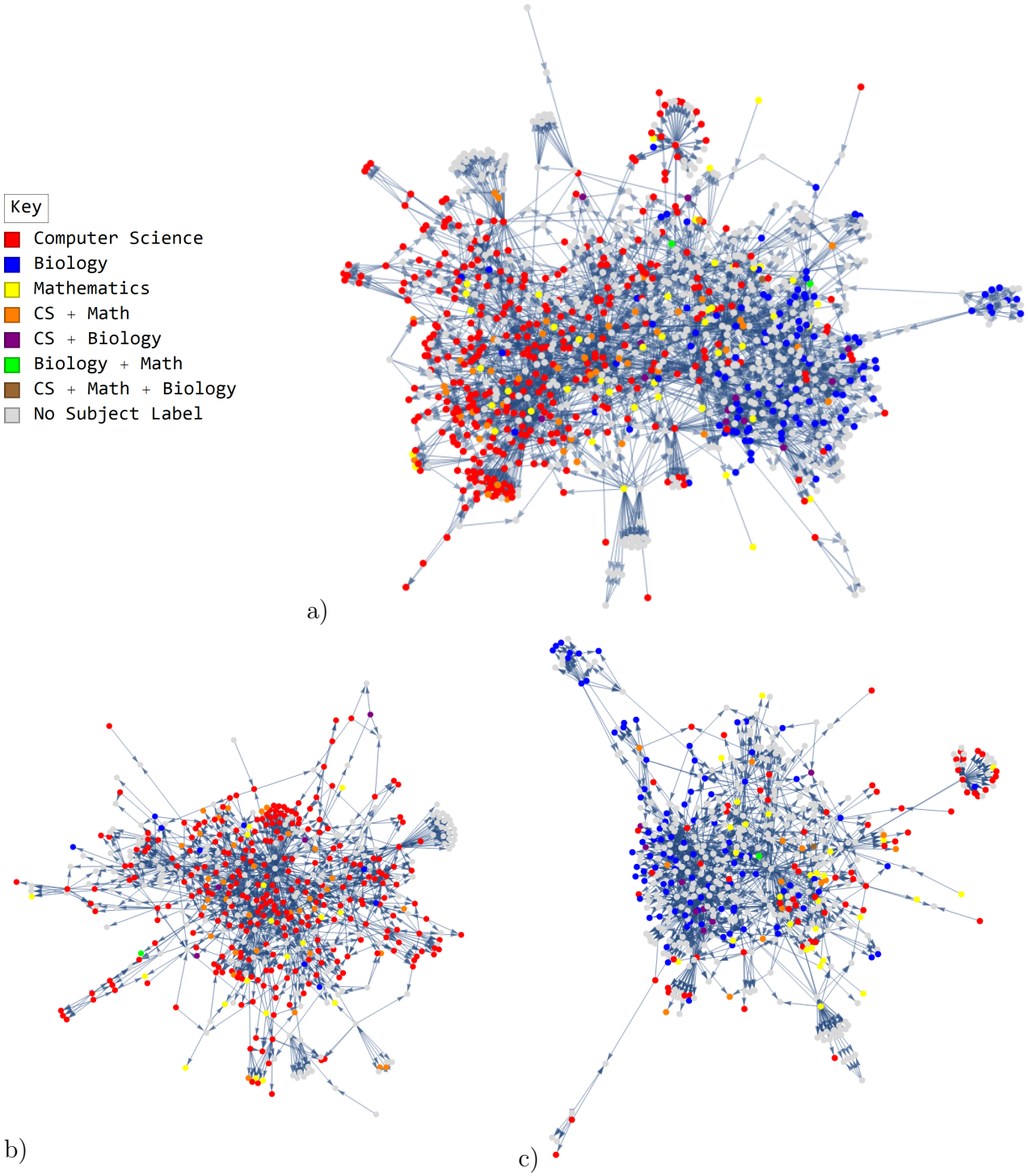


Figure 3.2: a) The pruned network G_p, and b)-c) two halves of its partition $G_p^{(1)}$ and $G_p^{(2)}$, with vertices colored according to their subject label.

	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 $G_p^{(2)}$.

of keywords in our search. As a result of this, the computer science network is more strongly 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, the subnetwork of the parent vertices, and the subnetwork of high centrality papers, as well as corresponding tables of vertex counts similar to Table 3.1, can be found in Appendix ?.

3.2.1 Assortativity results. We also 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. In Table 3.2, we calculate the assortativity with respect to

	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.

our tagged subjects, as well as various other network properties[4]. 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

[4]The biggest issue we face when calculating subject-based assortativity is that our vertices can belong to multiple categories, while the assortativity algorithm requires categories to be exclusive. To handle this, 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 isn't tagged as a certain subject or group of subjects, which was the approach for the rest of Table 3.2.

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. This makes sense, since (same journal? same subject tag?)

We then 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 The reading list also here's why reading things this way is awesome

In Section **??**, 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 this forms~~ our ~~initial reading~~ list. We will ultimately need to consult some other references, as this list does not include all key papers directly. However, that's less because we're adding stuff to what's considered most important, and more just that an important paper talks about stuff that's clearly important, but we need more detail on it, so we go fetch that reference.

Simply knowing why we are reading each paper, and in what ways it's considered more or less important in our collection, is incredibly helpful, especially when our goal is to read

a large number of papers in a relatively short amount of time, with the overall goal of classifying them into the overall shape of the field, and understanding everything in context. Like, knowing that somebody's important just because they cite everybody else, vs being cited a lot, not just overall, but inside the stuff we care about, that's so nice to guide reading. Not everything that's a hub is a survey, and not all the surveys are hubs. Etc.

The other really just ridiculously useful thing is looking at the subnetwork of the reading list papers themselves, so we can get the context. This is super extra nice for a survey, because the whole point is to be able to put the whole FIELD in context, and get an idea of the shape of it. So that is in Figure 3.3.

It's nice and well connected, only one paper isn't in the giant component. We didn't color by subject, because for some reason the papers in there are not well tagged at all, but you can still look at that in Figure A.2. Instead for right here we just colored the sides of the partition, and labeled them with the titles. You can pretty much tell which is which subject just by reading those. It follows the overall thing. Mostly bio on one side, CS is bigger. More of the important CS papers ended up being important overall, which makes sense, because it's bigger.

To make it even easier to put a paper we are reading in context, we can highlight its neighborhood, an example of which is shown in Figure 3.4. It helps with the importance thing, and then once you start reading, it's super nice to look at the context and connect it to the papers you've already looked at, or easily go back and check something real quick, to compare/contrast. Just really awesome. It's also cool to see the ones that are a bit interdisciplinary. You can sort of tell the ones that are interdisciplinary in the things they CITE, but it's also super interesting when a paper is CITED in an interdisciplinary way. It's not comprehensive, and it's a bit skewed by the bio side having some CS in it, but we (have not yet) made an appendix table where you can see which ones have a more even split across the two sides. Those are a combo of "that's actually CS, it's just on the other side", surveys, basics in the field, and a liiiiiiiiiittle bit of crossover between the CS alignment papers and

the bio alignment papers. Graph kernels also seem to give us some connection there.

Overall, doing it this way is much less overwhelming, and makes it much easier to avoid the thing where you only think something is the most important because it's what you know the best, or it stuck out to you while you were reading something. That seems to be a common problem with surveys. I don't know how people normally do it, and manual ways of looking at the connections are honestly probably less work than my way, but the nice thing is then I can SHOW it to you.

I noticed while reading that there really seem to be only two options that people go for when making surveys. They can either just collect a buttload of papers, and then present all the ideas equally, with some attempt to categorize methods and whatnot. That's how [6] does it. It's an awesome paper, but it's not beginner friendly. The other way is to collect a bunch, highlight a few of them, and sort of dump the rest in there somehow, which is what [8] does. This can sometimes help you get an idea of the methods, but if poorly executed, it's really really confusing, and a lot of work to grasp things, and you still don't really get a good sense of the shape of the field, because how on earth are you supposed to pick which references to read? It's a little better if you just highlight a few things and don't claim to be comprehensive, but that's got its own problems.

So, instead of that, we're going to highlight a few things generally, give you the basic intuition, tell you where to go look at them more, and highlight the differences in problem definition and such. Probably also a table somewhere which things are which categories, and which things are written in an accessible manner. Because I can't explain everything, and I shouldn't have to. That's a waste of my time and yours, it's better to go to the source, I'm just telling you where it is.

Overall, all of these reasons are what make me extra equipped to do an awesome job on the rest of this, and you should trust me with that. We're going to dive into the actual material now, yay.

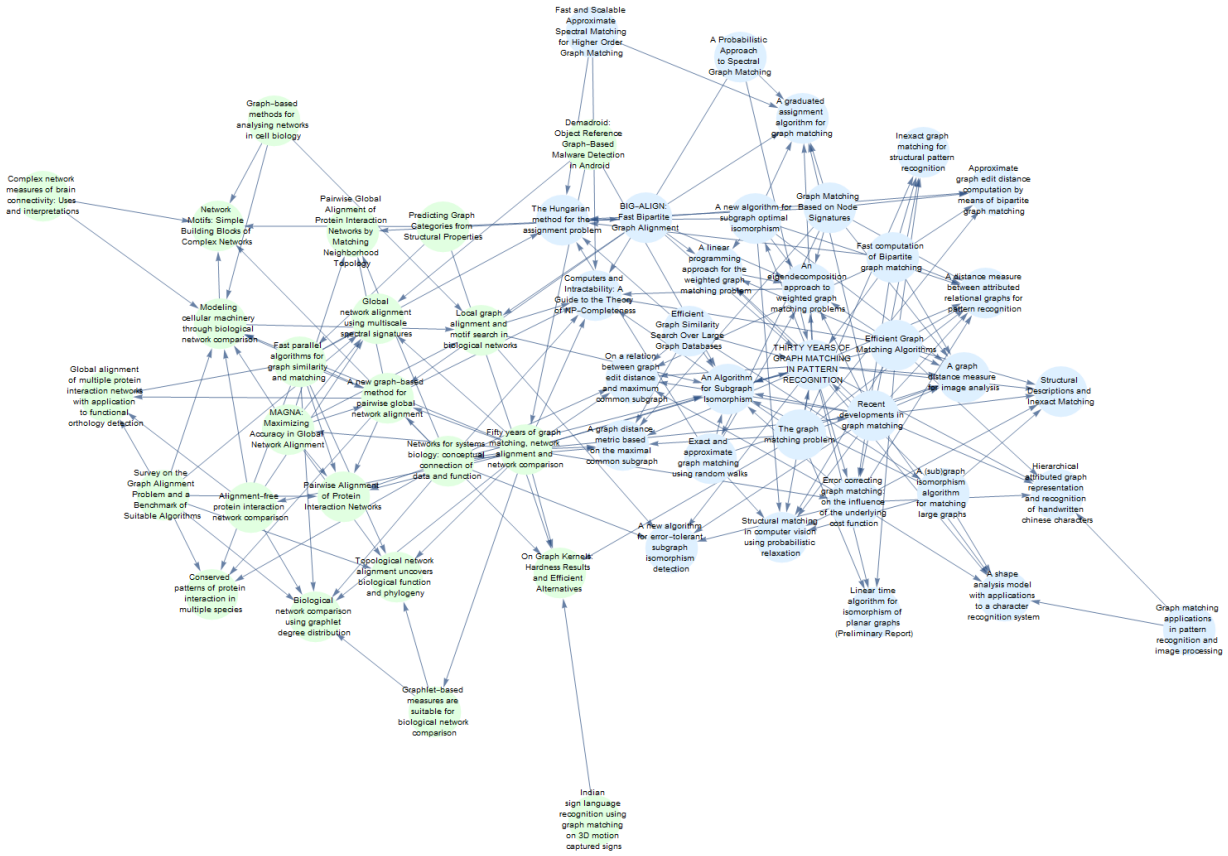


Figure 3.3: The subnetwork S of high centrality papers, as listed in Tables 2.2, 2.3, and 2.4. Green vertices are in group 1 (biology dominated) of the partition of G_p, and blue vertices are in group 2 (CS dominated).

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

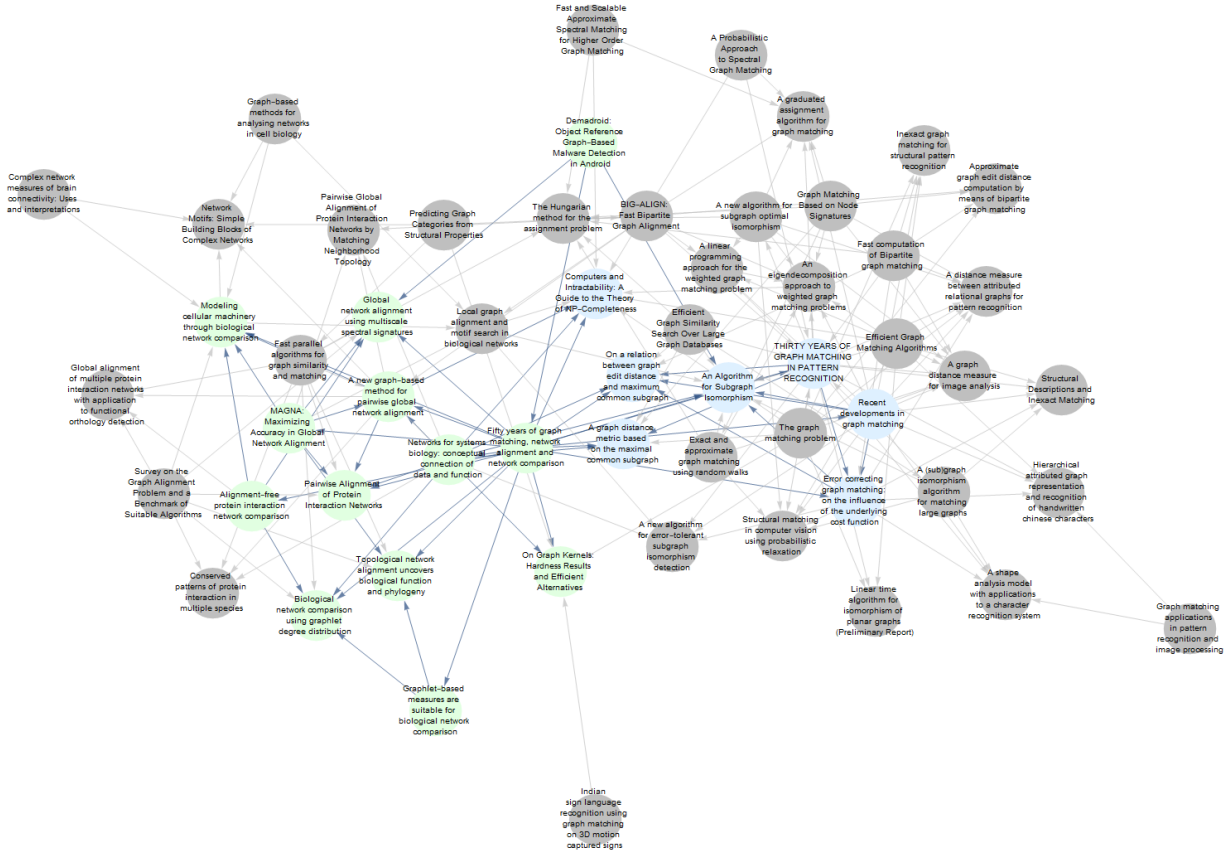


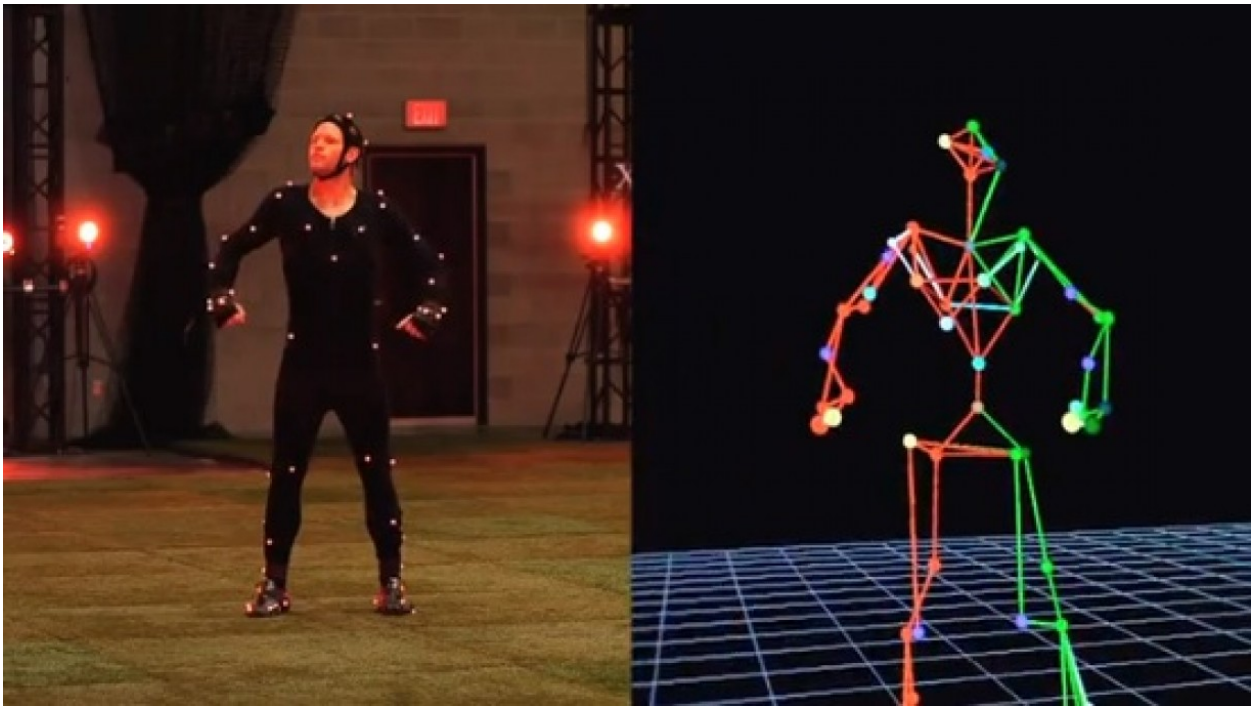
Figure 3.4: The subnetwork S of high centrality vertices, highlighting the neighborhood of "Fifty years of graph matching, network alignment, and network comparison".

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

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 translations, rotations, and mirroring, as well as other positional changes, they are well suited for this task.



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.

Applications in the area of computer vision include optical character recognition[16, 18], biometric identification[11, 7] and medical diagnostics[20], and 2D/3D object recognition in general[3]. In 2018, work has been published in the computer vision-related areas of Indian

sign language recognition[13, 12], spotting subgraphs (e.g. specific characters) in comic book images[14], and stacking MRI image slices[4]. A timeline with more comprehensive counts of papers appearing in various applicative areas in pattern recognition through 2002 can be found in "Thirty Years of Graph Matching in Pattern Recognition"[6].

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.

Note for me to put somewhere but I'm not sure where exactly. the cited papers besides the red book and newman are all parents

4.2 What is graph matching?

In the literature, the term "graph matching" is used far more often than it is explicitly defined, and 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 at all in survey papers. To make ~~matters worse~~, we can also address the question of finding a matching *in* a graph[25], 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 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[5]. Clearly a good taxonomy is needed.

In this section, we give an overview of these terms and summarize their distinctions.

4.2.1 ~~Miscellaneous~~ definitions. A decision problem (i.e. one which can be posed as a yes or no question) is in **NP** or **nondeterministic polynomial time** if the instances where the answer is "yes" can be verified or rejected in polynomial time[10, 27]. 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 [10, 28]. If a decision problem is both NP and NP-hard, it is **NP-complete**[10, 26].

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[24]. By contrast, a **subgraph** is simply a graph formed from a subset of the vertices and edges in the larger graph[22].

A **graph isomorphism** is a bijective mapping between the nodes of two graphs of the same size, which is **edge-preserving**; that is that is, if two nodes in the first graph are connected by an edge, they are mapped to two nodes in the second graph which are also connected by an edge[6]. 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[23].

A **subgraph isomorphism** is an edge-preserving injective mapping from the nodes of a smaller graph to the nodes of a larger graph. That is, there is an isomorphism between the smaller graph and some induced subgraph of the larger[6]. The decision problem of determining whether a graph contains a subgraph which is isomorphic to some smaller graph is known to be NP-complete[29].

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 [21]. 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.[2, 1]

	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[21].

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

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 nodes 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[15, 8]. For graph and subgraph isomorphism, this characterization is equivalent; either they are isomorphic/there is a subgraph in the larger which is isomorphic to the smaller, or they are not. Since the maximum common subgraph problem is edge preserved, 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[15].

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 nodes 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].

[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.

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 will theoretically be zero for two graphs which are isomorphic. An **optimal** algorithm, that is, 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 them significantly more expensive than their exact counterparts[6].

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 [6].

Mapping-seeking and non-mapping-seeking algorithms. Finally, we can draw the distinction of whether the algorithm seeks primarily to find a mapping between nodes (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 introduced in the final section of this chapter, and discussed fully in Chapter 5.

Remark. (THIS WHOLE REMARK IS WORD VOMIT) Livi's insistence that the mainstream approaches are edit distances, kernels, and embeddings. That's nonsense. It's more like edit distances, mapping-seeking strategies (alignment), and non-mapping seeking strategies (most kernels, graphlets, embeddings).

Livi [15] makes a big deal out of graph kernels, including them as one of his main approach categories, but I think that's just, like, his opinion, man. I looked at his references he cites for them, there's ten of them, and they're pretty much all cited only by him in the dataset. They're newer than "thirty years", generally, so they're not in there. Based on some brief

reading, I think they fall into the category of the alignment strategies, where you do a node similarity and then turn that into a matching (or into a kernel). He's like "in the actual scientific literature, we can clearly distinguish three mainstream approaches", but that's stupid, and a good reason why my way of doing surveys is better.

Is the continuous part of the kernel definition really that hard? What your open sets gonna be on the space of graphs? Continuous with respect to the discrete topology? Stupid.

I'm fairly sure graph kernels aren't more fully investigated, because any kernel on which we could prove nice enough results to use nice kernel techniques would not be polynomial time. Although honestly, the real reason it hasn't turned up for me is because "graph kernel" wasn't one of my ten search terms, I'm guessing. I still am willing to bet that it's a bit more niche, and more along the lines of the alignment strategies–turn things into matrices, node similarities, combine those in clever ways.

Just frame the graph embedding idea as analagous to the graphlets. Definitely bring up them all in the camp of "non-mapping-seeking", just that kernels are stricter. Like how a metric is stricter than a generic mapping into $[0, 1]$. Mention that they bring those up here, mention why you didn't see more of them, that you think it's niche, and from what you can tell they're better framed in terms of the bio camps of "graphlet style" and "alignment style". Useful for pattern recognition as like a training set thing, not as broadly represented in my dataset.

END OF REMARK

4.3 Exact strategies and Edit Distances

Honestly, I'm not sure what to do with this section at this point. I don't really want to do a whole long survey thing on search space pruning and edit distance strategies, but I feel like I should include SOMETHING. Can I just make a short little section pointing them where to look? Could use some discussion on it. The whole rest of everything is just notes.

	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. *I'm sure there are optimal methods out there, but we didn't find any, and if they exist they're certainly not polynomial time.

4.3.1 pruning the search space (isomorphism). For image recognition, the task at hand is **inexact** graph matching; we are looking for model graphs which are *similar* to an input graph, and we cannot generally hope to find a perfect match. However, we believe a discussion of different types of graph matching is best served by beginning with a discussion of graph **isomorphism**, as all other types can be considered to be a weakening of the conditions for graph isomorphism.

This one goes first, because it naturally fits with isomorphism, which is good because we want to start with the strictest problem formulation and work our way down the weakening of it. Which is how the timeline goes, anyway.

4.3.2 Edit distances (inexact/error tolerant?). I think edit distances are like a natural weakening of the problem, at least compared to alignment. A good next step time-wise, too.

4.4 Alignment

This goes last, because it's the most closely tied to bio, and also goes along with the theme of broadening the scope of the problem, maybe?

4.4.1 Topological node similarity.

4.4.2 Weight matching vs. bipartite graph matching and the Hungarian algorithm.

It's important that they understand about that, since most of the readings sort of take that connection for granted, and it's confusing.

Chapter 5. Systems Biology

5.1 Types of biological networks and goals of analysis

Especially protein-protein interaction networks

5.2 Network statistics leading into graphlets and such

5.2.1 Introduction to univariate statistics via degree distributions.

That's a good way to introduce the idea of local vs. global statistics–reference all the GLOBAL statistics we already talked about

5.2.2 Applications of univariate statistics.

5.2.3 Motifs and graphlets are a generalization of degree distributions.

5.2.4 They are important both because they can be used to create comparison metrics, and because random models for networks are important.

5.3 Alignment Overview

5.3.1 Why do we want to do it?. Analogy to sequence alignment goes HERE.

5.3.2 What is it exactly?. What is the actual definition, compared to how we just defined it for CS? Also, external information is very important! Not just topological similarity! We need it very badly!

Figure 5.1: Steal figure 1 from IsoRank 2007 intro? (last one in the global alignment stack)

5.3.3 Local vs. global.

Table 5.1: nice table of the named local and global algorithms, their year, their source number in the reference list, their similarity strategy, and their search strategy.

5.3.4 Two steps of the process: similarity matrix + searching. Because I already talked about this in the CS chapter, I can just do it comparatively here.

why is bio smaller? look what we got inside of CS. It is computer vision and natural language processing. That's pretty cool. What does the intersection look like? Biometrics and stuff. Bio people like image processing too! It is cool.

What problems are we trying to solve when we're working with metabolic networks? Why do we care about looking up proteins? figure out what stuff does? Why do we care about protein interaction How is DNA sequence alignment network similarity? Is it just that

all these microRNA papers cite those kinds of papers a lot?

Mapping the brain to figure out what's normal and what's important and what does what? Is that really network similarity? Community detection requires a notion of similarity.

based on a quick glance through titles, bio applications seem to be protein folding (and molecular similarity? chemical structures), metabolic interaction networks (duh) gene stuff, something something microRNA (same thing?), brain networks, SOCIAL networks, protein database search (should be similar to the graph matching ones where it's trying to find an image in a large database)

Predictions: social network and metabolic network relatively similar approach? Large networks of interactions. Molecular structure on a small scale should be like graph matching?

The main two kinds of applications seem to be computer vision and natural language processing. Both of those make sense. Unique identifiers, grading, image processing, large database search, e.g. handwriting and fingerprint/facial/iris recognition

Computer vision: fingerprint classification (lots of those, especially older ones), shape matching, pulling objects from an image,

Natural language processing: semantic relations, obviously

Niche things: malware classification.

Types of problems we're trying to use network similarity for?.

- Searching for things in a large database of small graphs (fingerprint classification, protein search, facial recognition). Nearest neighbor type thing. Unique identifiers is really more of a vertex similarity thing, but oh well.
- Classify things as normal or anomalous (malware, cancer, trajectories, grammar,)

Chapter 8. Glossary (All background beyond the very basics for the introduction)

Appendix A. Appendices

- Table with the references, what category they're in, and short note?

Table A.1: table of the reading list papers according to my sticky note categories

A.1 Additional Citation Network Figures and Tables

A.2 Code-related

A.3 Subject Tagging Keywords

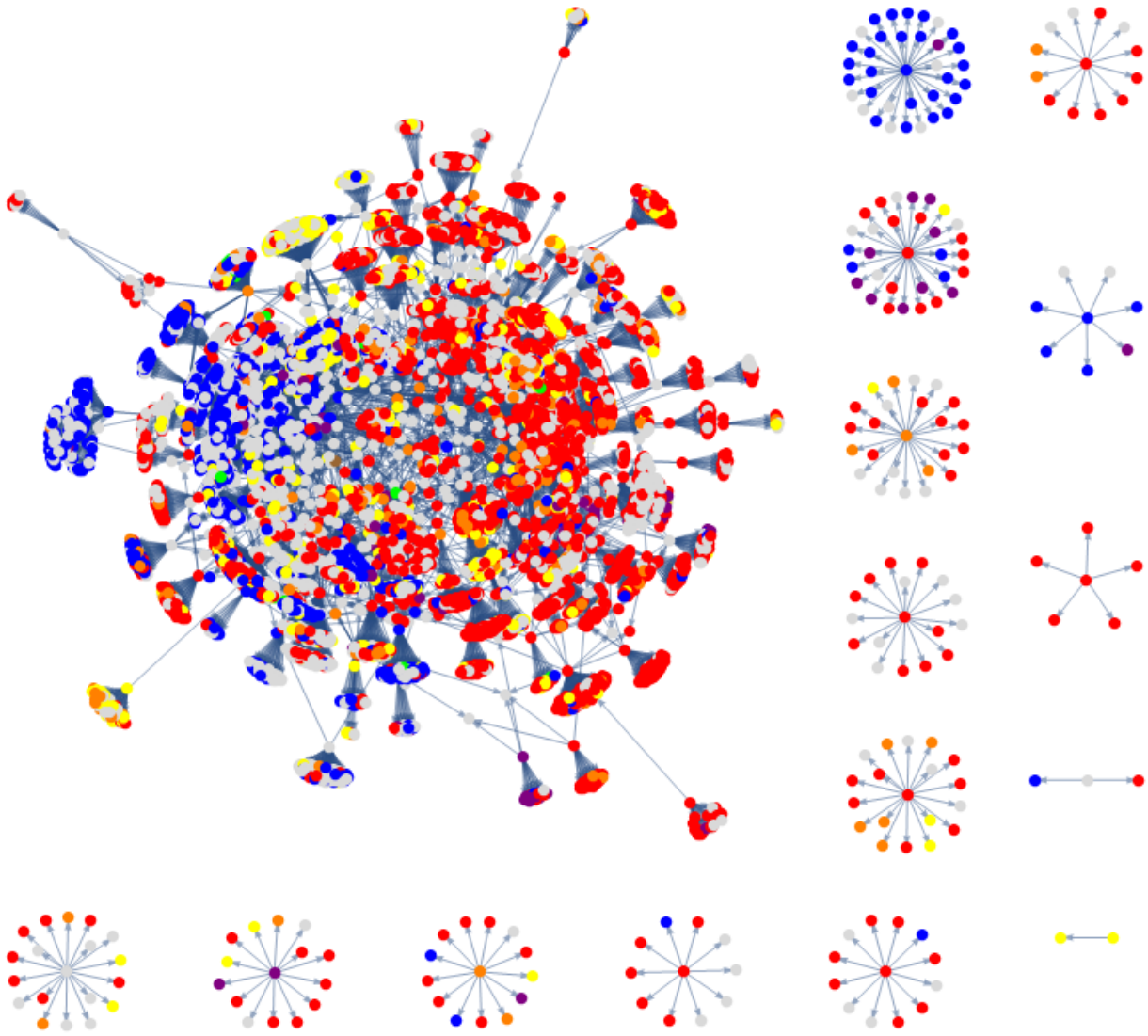


Figure A.1: The full network G_p, with vertices colored according to their subject label as in Figure 3.2.

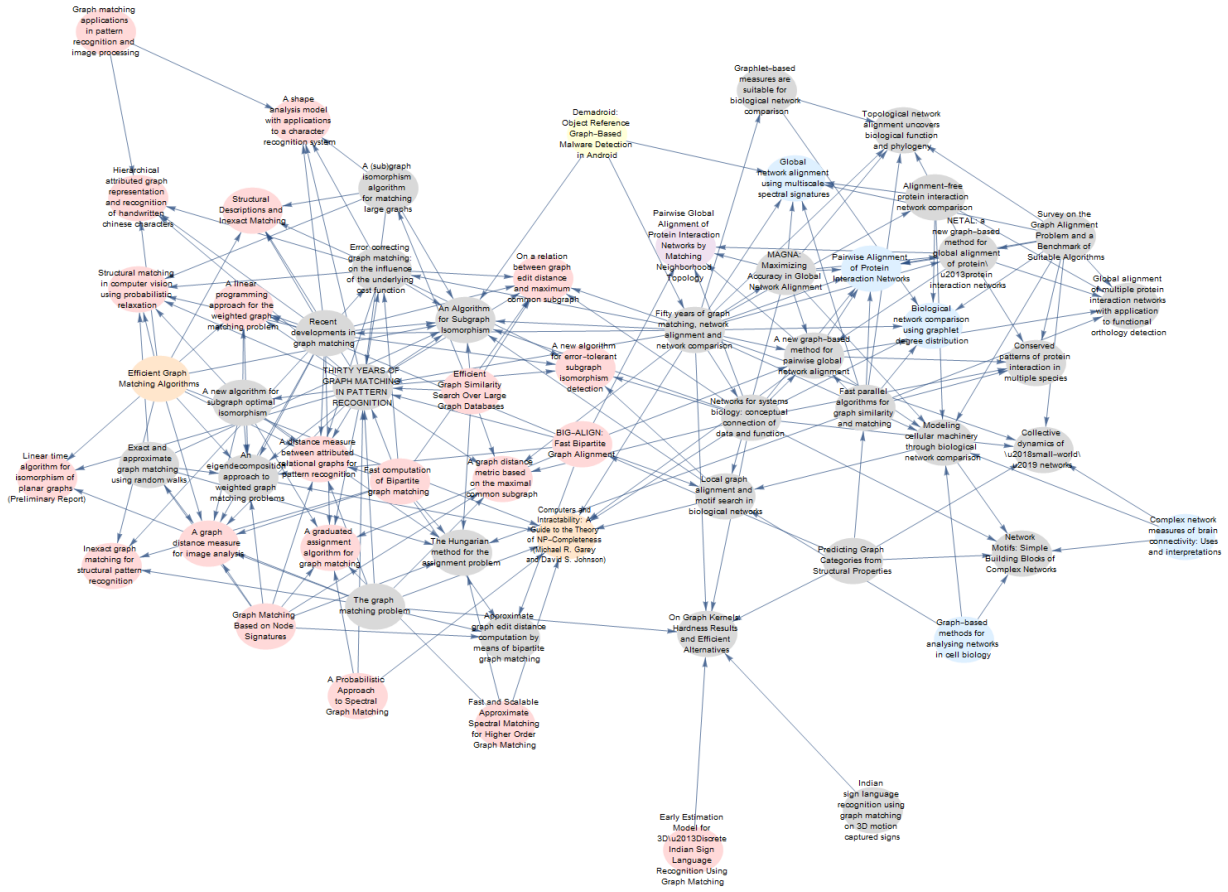


Figure A.2: The subnetwork S of high centrality papers, as listed in Tables 2.2, 2.3, and 2.4, with vertices colored according to their subject label. Grey vertices are unlabeled, pink is computer science, blue is biology, yellow is math, orange is both math and computer science, and purple is both computer science and biology.

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

Figure A.3: 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.

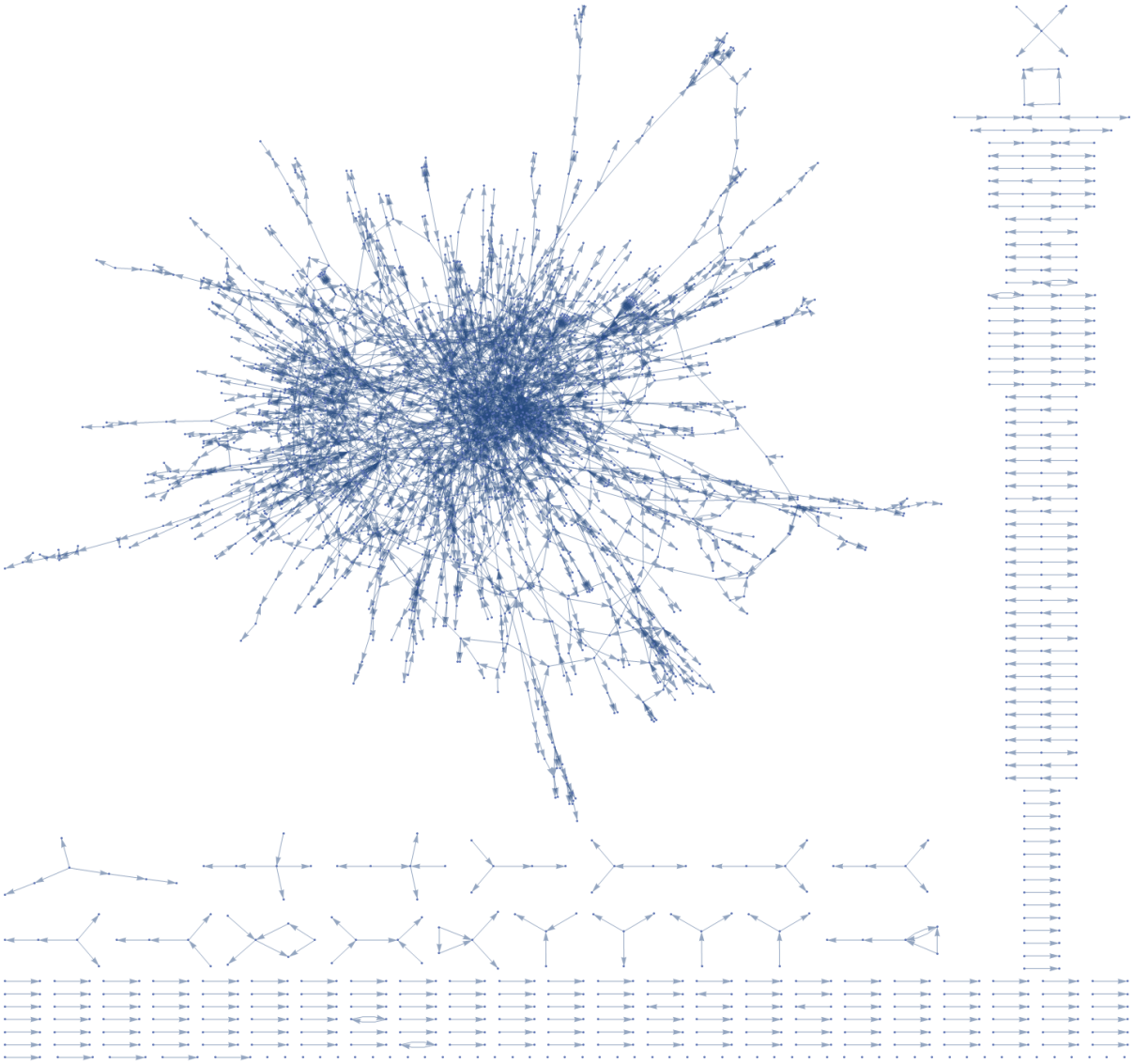


Figure A.4: The zewail citation network dataset used in our Table 2.1 comparison to G. Note the high number of connected components, and low number of children per parent, in contrast to the "blooming" behavior in G created by the inclusion of ALL child references from each parent paper.

Initial loading of G, sciMet, and zewail GML files

```
G = Import["citation_network.gml"]
sciMet = Import["sciMet_dataset.gml"]
zewail = Import["zewail_dataset.gml"]
```

Creating the pruned network G_p.

```
parents = Position[VertexOutDegree[G], _?(# > 0 &)] // Flatten;
popular = Position[VertexInDegree[G], _?(# > 1 &)] // Flatten;
pruned = Subgraph[G, Union[parents, popular], Options[G]];
Gp = Subgraph[pruned, WeaklyConnectedComponents[pruned][[1]],
    Options[pruned]]
```

Creating the random network R.

```
R = RandomGraph[{VertexCount[G], EdgeCount[G]}, DirectedEdges -> True]
```

Creating the random network R_d with the same degree sequences as G.

```
outStubs = VertexOutDegree[G];
inStubs = VertexInDegree[G];
vertexlist = Range[VertexCount[G]];
edgelist = {};
For[i = 1, i <= EdgeCount[g], i++,
 source = RandomSample[outStubs -> vertexlist, 1][[1]];
 target = RandomSample[inStubs -> vertexlist, 1][[1]];
 outStubs[[source]] -= 1;
 inStubs[[target]] -= 1;
 AppendTo[edgelist, source -> target];]
Rd = Graph[vertexlist, edgelist];
```

Figure A.5: The Mathematica code used to create the networks analyzed in Table 2.1.

	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.2: 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 oauth2client
The `defaultdict` datatype	collections
Interfacing my own modules with Jupyter notebooks	importlib

Table A.3: Python packages used for the project.
*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.

Bibliography

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

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

[18] 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.

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

[20] 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.

[21] 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].

[22] 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].

[23] 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].

[24] 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].

[25] 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].

[26] 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].

[27] 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].

[28] 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].

[29] 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].

Index