12-2015

Multipartite Graph Algorithms for the Analysis of Heterogeneous Data

Charles Alexander Phillips
University of Tennessee - Knoxville, cphill25@vols.utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Charles Alexander Phillips entitled "Multipartite Graph Algorithms for the Analysis of Heterogeneous Data." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Michael A. Langston, Major Professor

We have read this dissertation and recommend its acceptance:

Bruce J. MacLennon, Brynn H. Voy, David J. Icove

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Multipartite Graph Algorithms for the Analysis of Heterogeneous Data

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Charles Alexander Phillips
December 2015

Acknowledgements

In the course of my education and research I have had the good fortune to cross paths with many bright and dedicated people. I extend my gratitude to many of them here, although I am certain the list is not complete. First I would like to thank my advisor, Dr. Michael A. Langston, for his guidance, patience and above all the example he sets for high standards in scientific research and work ethics. My special thanks go out to those who served on my dissertation committee: Drs. David Icove, Bruce MacLennan, Lynne Parker and Brynn Voy. Former and present students I have worked with as part of Dr. Langston's research team here at the University of Tennessee include John Eblen, Ron Hagan, Jeremy Jay, Jordan Lefebvre, Allan Lu, Sudhir Naswa, Clinton Nolan, Andy Perkins, Gary Rogers, Kai Wang, Dinesh Weerapurage and Yun Zhang. Research Collaborators include Erich Baker, Jason Bubier, Elissa Chesler, Frank Dehne, Dan Goldowitz, Mike Miles and Aaron Wolen. My thanks go to Suzanne Baktash for her encouragement, helpfulness and support. Other professors at UT who I have been fortunate to collaborate with include Drs. Arnold Saxton and Meg Staton. My appreciation goes to instructors at Moberly Area Community College, where I completed my associate degree, and Columbia College, where I did my bachelor's degree, for helping to fan the spark of my interest in computer science and encouraging me to pursue a graduate degree. These instructors include David Heise, Yihsiang Liow, David Pence and Lawrence West. And last but not least, my gratitude and appreciation goes to my family: my sister Lisa, brothers Chris and Mike, stepmother Sylvia, and especially my father, Alex, whose support and encouragement through the years have been beyond price.

Abstract

The explosive growth in the rate of data generation in recent years threatens to outpace the growth in computer power, motivating the need for new, scalable algorithms and big data analytic techniques. No field may be more emblematic of this data deluge than the life sciences, where technologies such as high-throughput mRNA arrays and next generation genome sequencing are routinely used to generate datasets of extreme scale. Data from experiments in genomics, transcriptomics, metabolomics and proteomics are continuously being added to existing repositories. A goal of exploratory analysis of such omics data is to illuminate the functions and relationships of biomolecules within an organism. This dissertation describes the design, implementation and application of graph algorithms, with the goal of seeking dense structure in data derived from omics experiments in order to detect latent associations between often heterogeneous entities, such as genes, diseases and phenotypes. Exact combinatorial solutions are developed and implemented, rather than relying on approximations or heuristics, even when problems are exceedingly large and/or difficult. Datasets on which the algorithms are applied include time series transcriptomic data from an experiment on the developing mouse cerebellum, gene expression data measuring acute ethanol response in the prefrontal cortex, and the analysis of a predicted protein-protein interaction network. A bipartite graph model is used to integrate heterogeneous data types, such as genes with phenotypes and microbes with mouse strains. The techniques are then extended to a multipartite algorithm to enumerate dense substructure in multipartite graphs, constructed using data from three or more heterogeneous sources, with applications to functional genomics. Several new theoretical results are given regarding multipartite graphs and the multipartite enumeration algorithm. In all cases, practical implementations are demonstrated to expand the frontier of computational feasibility.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BL6	Black 6 inbred mouse strain, aka C57BL/6J
BK	Bron-Kerbosch
BK-K	Bron-Kerbosch *k*-partite
CbGRiTS	Cerebellar Gene Regulation in Time and Space
DBA	DBA/2J inbred mouse strain, aka D2
EMR	Electronic Medical Records
ES	Essential Set
eQTL	Expression Quantitative Trait Locus
FDR	False Discovery Rate
FPT	Fixed-Parameter Tractable/Tractability
GEO	Gene Expression Omnibus
GO	Gene Ontology
iMBEA	Improved Maximal Biclique Enumeration Algorithm
ISH	In-Situ Hybridization
KO	Knockout (Gene Knockout)
LRS	Likelihood Ratio Statistic
MBEA	Maximal Biclique Enumeration Algorithm
MC	Maximum Clique
MCC	Maximum Clique Cover
MCE	Maximum Clique Enumeration
MCF	Maximum Clique Finder
MICA	Modular Input Consensus Algorithm
mRNA	Messenger RNA
PPI	Protein-Protein Interaction
QTL	Quantitative Trait Locus

Chapter 1 Introduction and Background

Moore's law observes that the processing power of computers has roughly doubled every two years since the 1970's. But in recent years the rate has begun to slow, and the slowing is expected to continue as microchip technology approaches fundamental physical limits. At the same time, growth in the amount of data generated in recent years has outpaced the increase in computational power. This trend also is expected to continue. In order to keep pace with the gap between computational power and data growth, there is a need for the continual development of new algorithms and analytic methods.

Nowhere is the growth of data more apparent than in the biological sciences, especially in genetics and genomics. Technologies that led to the sequencing of the human genome, among other breakthroughs, have continued to advance. Technologies such as microarrays, RNA-seq and mass spectrometry produce ever-increasing amount of data. A wealth of experimental data now resides in publicly available repositories, the experiments' authors applying only a comparatively small number of analytic techniques focused on but a few hypotheses. Such data almost certainly contains undiscovered knowledge, only awaiting the application of the right algorithms and techniques.

This dissertation focuses on the development and application of graph-theoretical algorithms to large-scale data, such as data generated by current genetics and genomics technologies. The algorithms have the common theme of seeking dense structure in graphs, dense structure being loosely defined as highly interconnected sets of nodes. Density-related problems arise in a host of research fields, as diverse as computational molecular biology [1], telecommunications [2], natural language processing [3], social network analysis [4], transportation [5, 6], operations research [7], chemistry [8], drug discovery [9], phylogeny [10] and ad hoc networking [11]. Such dense structure can be used to infer co-associated sets of entities. The algorithms have all been implemented, tested, and applied to experimental data. Emphasis has been placed on practical implementations, since the algorithms must scale to the growing size of biological datasets. The algorithms described and developed here are targeted to biological problems, but since they are designed using an abstract graph model, they should have applications well beyond the specific problem by which they were inspired.

Another theme running through this thesis is the use of exact algorithms, even for combinatorial problems generally considered intractable. A problem's *NP*-completeness is often stated as a reason why approximate or probabilistic algorithms or heuristic solutions are needed. The usual philosophy at the Langston

Lab, however, is to eschew such thinking. Our general view is that, given the time and expense of data collection, one should not cut corners on data analysis methods without first attempting to find exact optimal solutions to the problem at hand.

Most of the major pieces of this dissertation consist of previously published papers that I co-authored as part of the Langston Lab and to which I contributed original work. The different publications are brought together and organized here into three broad categories: algorithms for general graphs, algorithms for bipartite graphs and algorithms for multipartite graphs. Each of the major sections of chapters 2 and 3 (sections 2.1, 2.2, 2.3, 2.4, 3.1 and 3.2) represent one distinct publication. A preliminary version of chapter 4 has been published as well. Often, our lab's contribution to a publication focused on some biological problem has been to provide novel data analytics. As such, I have reproduced here only those parts of these publications that describe our contributions or that are necessary to provide context. To avoid redundancy, much of the background and introduction from the publications has been combined and placed in this introductory chapter.

1.1 Definitions, Notation and Preliminaries

A graph is an abstract representation consisting of a set of vertices, which represent objects, and a set of edges, which represent association between pairs of objects. A graph is also called a network; vertices are also called nodes; and edges are also called arcs, links or connections. In this work we consider only simple, unweighted, undirected graphs. Weighted graphs are only discussed in the context of converting them to unweighted graphs via a thresholding procedure on the edge weights.

We use the following definitions and notation. The number of vertices in a graph is the *order* of the graph, sometimes referred to as the graph's size. For a graph G, $V(G)$ denotes the set of vertices and $E(G)$ denotes the set of edges. Often just V and E are used when no ambiguity results. The cardinality of V (number of vertices in the graph) is denoted by $|V|$ or, when no ambiguity results, by n. The cardinality of E (number of edges in the graph) is denoted by $|E|$. Edges are denoted by their endpoints; for example uv and (u, v) both denote an edge between vertices u and v. The neighborhood of a vertex $v \in V(G)$ is defined by $N_G(v) = \{ u \in V(G) : uv \in E \}$, and is also written as $N(v)$ when no ambiguity results. Note that the cardinality of $N(v)$ is the degree of v.

A *clique*, or *complete* graph, is a graph with all possible edges. An *independent set* is a graph with no edges. A *vertex cover* is a set of vertices in a graph such that edge in the graph has at least one endpoint in the set. A *maximum* clique is a clique of largest size in a graph. Similarly, a maximum independent set is an independent set of largest size and a minimum vertex cover is a vertex cover of smallest size. A

maximal clique is a clique that is not properly contained in another clique; e.g. no vertex can be added to form a larger clique. A graph is k-*partite* if it can be partitioned into k disjoint independent sets, called *partite sets*. A k-partite graph is also called *multipartite*, typically only when $k \geq 3$. Edges in a k-partite graph are *interpartite*, having endpoints in different partite sets. By definition, a k-partite graph has no *intrapartite* edges, edges between vertices in the same partite set. (Nevertheless we will discuss intrapartite edges in the context of our algorithm in chapter 4.) A k-*partite clique*, or complete k-partite graph, is a k-partite graph with all possible interpartite edges. A maximal k-partite clique is a k-partite clique to which no vertex can be added to form a larger k-partite clique. A 2-partite graph is *bipartite*, and a 2-partite clique is a *biclique.* Similarly, a 3-partite graph is *tripartite*, and a 3-partite clique is a *triclique*. Rather than extend the Latin nomenclature to quadricliques, quinquecliques, and so forth, for readability we use numerals for any $k > 3$, i.e. 4-partite clique, 5-partite clique, etc. We will assume that by definition a k-partite clique must include at least one vertex from each partite set. The density of a graph is the ratio of the number of edges to the number of possible edges; in k-partite graphs, intrapartite edges are not considered possible edges. Sometimes density is reported as a percentage rather than a proportion.

When considering maximum k-partite cliques, we must distinguish between vertex-maximum and edge-maximum. The distinction is only relevant for maximum, not maximal k-partite cliques. Whether a k-partite clique is maximal depends only on whether or not it can be extended by inclusion of another vertex. Just as there can be more than one maximum clique in a graph, there can be more than one maximum k-partite clique in a k-partite graph, both for the edge maximum and vertex maximum varieties.

A complete graph on n vertices is denoted by K_n. A complete bipartite graph is denoted by $K_{m,n} = (U,V)$ where U and V are the two partite sets of G. A complete k-partite graph is denoted by $K_{x_1,\ldots,x_k} = (V_1, \ldots, V_k, E)$, where $V_1, \ldots, V_k$ are partite sets with respective cardinalities $x_1, \ldots, x_k$.

A k-partite graph is *balanced* if the partite sets differ in number of vertices by at most one. Balanced k-partite graphs are also called Turán graphs. Turán studied such graphs in the context of extremal graph theory, since they have the maximum number of edges possible for a graph that does not contain a (k-1)-clique, as stated by Turán's theorem [12, 13]. As we shall show, balanced k-partite graphs are also useful for proving bounds we describe here.

A more comprehensive source of graph theory terminology and notation can be found in [14].

1.2 Omics Data

The word *omics* refers to several research areas in biology ending with the same suffix. Some of the best known examples include genomics, transcriptomics, proteomics and metabolomics. These areas are characterized by an abundance of large-scale experimental data measuring particular aspects of biomolecular pathways.

Transcriptomic data measure the abundance and types of RNA molecules. Until recently the most often used transcriptomic technology was the messenger RNA (mRNA) microarray, which measures the relative abundance of tens of thousands of different mRNA sequences simultaneously in a tissue sample. Each sequence is mapped to the particular gene that coded it, and for this reason mRNA microarray data is referred to as gene expression data. Since the advent of expression microarrays roughly two decades ago, a plethora of such data has become available. RNA microarrays are gradually being supplanted by "next-generation sequencing," or RNA-seq, which measures actual counts of RNA sequences in a sample. When discussing mRNA data, we often use probesets, probes and genes synonymously, though they are not technically the same.

Proteomics studies the structure, behavior and interaction of proteins at a large scale. The two most common technologies for detecting proteins are mass spectrometry and immunoassays. There are several publicly available databases containing protein information, including known interactions between proteins. These interactions can be modeled as protein-protein interaction (PPI) networks, which are well-suited for analysis using graph algorithms.

Metabolomics studies metabolites, or small molecules involved in biomolecular pathways. Like proteomics, it uses mass spectrometry to measure the presence and abundance of molecules and molecular fragments.

The methods presented in this work are typically motivated by specific problems arising in the context of the particular set of data to which they are applied. But the algorithms can apply to other types of data in a variety of domains, as long as the data can be modeled as the appropriate type of graph.

1.3 Constructing Graphs from High-Throughput Data

Since the algorithms described in this work are restricted to simple, unweighted, undirected graphs, the first goal when presented with data is to create such a graph. To use a graph model, we need the concept of vertices (nodes) and edges (connections). The entities represented by vertices depend on the data, but are usually straightforward to interpret. In gene expression data, for instance, the vertices are genes. In proteomics data, the vertices are proteins. In functional

genomics, the vertices may represent samples, diseases, metabolites, phenotypes and a host of other entities. To place edges between vertices, we need some meaning for the edge, some way in which entities that the two vertices represent are associated. In graphs with proteins as vertices, for instance, edges may be placed between every pair of proteins that are known to interact. If genes are the vertices, then the edges may connect pairs of genes that have functional similarity or have highly correlated expression across samples. If there is a similarity metric from which one can compute a similarity score, then that score can be interpreted as an edge weight. A thresholding procedure can then select only those edges above (or below) a given threshold to create an unweighted graph.

1.3.1 Similarity Metrics

There are many different metrics for computing similarity. The choice of metric depends on the entities being modeled, the nature of the data and the goal of the analysis. Pearson correlation, for example, can be used to find linear dependence between two entities with quantitative measurements across common conditions such as samples, time points or dosages. Mutual information can be used on data with categorical measurements or when non-linear relationships are sought. Jaccard similarity is used to measure similarity between two sets. Any quantitative measure of difference or distance can of course be converted into a similarity measure simply by taking the inverse, and vice versa. A Euclidean distance d, for example, can be converted into a proximity score of $1/d$. Table 1 lists some common similarity metrics and their uses.

1.3.2 Thresholding

Once we have a similarity score between each pair of vertices, we can create an unweighted graph by selecting a threshold, some value of the similarity score. When a pair of vertices has similarity at or above the threshold, an edge is placed between the two vertices. Otherwise, no edge is placed.

The selection of an appropriate threshold is an ongoing research area. The selection is somewhat analogous to the selection of p-value at which to call a result statistically significant, in that the goal is to strike a reasonable balance between false positives and false negatives. One technique for selecting a threshold is to choose the highest threshold that places an edge between all pairs of vertices with known or expected interactions. Similarly, one can select a threshold so that one or more pairs of vertices with known interactions appear in the same paraclique. Both methods are fraught with potential problems, however, in part because of the reliance on a few

Table 1. Common similarity metrics.

Metric	*Measures*	*Notes/Uses*
Pearson Correlation	Linear dependence	Many
Spearman Correlation	Linear dependence of ranks	Resistant to outliers
Mutual Information	Non-linear dependence	Categorical data
Cosine Similarity	Similarity in vector space	Document comparison
Euclidean Distance	Distance in Cartesian space	Straight line distance
Orthodromic Distance	Distance on a sphere	Geocoding
Jaccard Similarity	Similarity between sets	Set-set comparison
Hamming Distance	Difference between equal length strings	Error correction
Levenshtein Distance	Difference between strings, where insertions and deletions are expected	Sequence comparison
Known Association	Molecular interactions, known pathways	Protein-protein interactions, etc.

interactions being accurately reflected in the data. Misconceptions about which interactions ought to be found, experimental confounds and noise from a variety of sources can all render such methods unreliable. Data-driven methods are thus usually preferred. Such methods include spectral graph clustering [15] and fitting to a scale-free topology [16]. For a comparison of six threshold selection methods for gene expression data, see [17].

1.4 The Quest for Dense Subgraphs

Algorithms for finding dense subgraphs are a subclass of clustering algorithms. As such, they can have a variety of characteristics. Some produce disjoint subgraphs, where others produce overlapping subgraphs. Some assign all vertices in a graph to some cluster; others can leave some vertices unassigned.

A clique is the densest possible subgraph, since it contains all edges. Therefore cliques are natural structures to seek when searching for dense subgraphs. Clustering techniques based on graph-theoretical algorithms have numerous advantages over traditional methods. Clique-centric strategies tend to be especially effective. They are resistant to false positives, and naturally accommodate pleiotropism through overlap. Algorithms presented in this work to seek dense subgraphs are ultimately clique-based. In computational biology, one needs to look no farther than PubMed to gauge clique's utility in a variety of applications. A

notable example is the search for putative molecular response networks in high-throughput biological data. Popular clique-centric tools include clique community algorithms for clustering [18] and paraclique-based methods for QTL analysis and noise abatement [19, 20]. Clique-based clustering algorithms have been shown to be superior to other clustering methods [21]. Besides cliques, other types of dense subgraphs that may be sought include *k*-clique communities [18] and *k*-plex's [22].

Problems related to clique elucidation are among the most widely-studied issues in graph theory. They are also among the hardest. Simply determining whether a graph has a clique of a given size was one of Karp's original 21 *NP*-complete problems [23] and is known to be difficult even among *NP*-complete problems, being W[1]-complete [24]. Even approximating the maximum clique size is problematic; the maximum clique size is not polynomial time approximable to within a factor of $n^{1-\varepsilon}$ for any $\varepsilon > 0$ unless co*NP* = *NP* [25].

1.4.1 Maximum Clique

Finding a maximum clique is a notoriously difficult combinatorial problem. Although classically formulated as an *NP*-complete decision problem [26], where one is merely asked to determine the existence of a certain size clique, the search and optimization formulations are probably most often encountered in practice, where one is asked to find a clique of given size and largest size respectively.

Fortunately, real-world graphs tend to be sparse and have inherent topological structure that makes their solution feasible with state-of-the-art algorithmic implementations [27, 28]. The algorithms can be sped up with techniques inspired by fixed-parameter tractability (FPT) [24, 29] such as common neighbor preprocessing and color preprocessing [30].

A maximum clique is particularly useful in our work on graphs derived from biological datasets. It provides a dense core that can be extended to produce plausible biological networks [31]. Other biological applications include the thresholding of normalized microarray data [15, 17], searching for common *cis*-regulatory elements [32], and solving the compatibility problem in phylogeny [33]. See [34] for a survey of additional applications of maximum clique.

1.4.2 Maximal Clique Enumeration

The set of maximal cliques forms a natural overlapping clustering of a graph with the most rigid requirements, namely that all edges be present within each cluster. Current maximal clique enumeration algorithms can be classified into two general types: iterative enumeration (breadth-first traversal of a search tree) and backtracking (depth-first traversal of a search tree). Iterative enumeration

algorithms, such as the method suggested by Kose *et al* [35], enumerate all cliques of size k at each stage, test each one for maximality, then use the remaining cliques of size k to build cliques of size $k + 1$. The process is typically initialized for $k = 3$ by enumerating all vertex subsets of size 3 and testing for connectivity. In practice, such an approach can have staggering memory requirements, because all cliques of a given size must be retained at each step. In [36], this approach is improved by using efficient bitwise operations to prune the number of cliques that must be saved. Nevertheless, storage needs can be excessive, since all maximal cliques of one size must still be made available before moving on to the next larger size. Figure 1 shows the number of maximal cliques of each size in a fairly typical graph constructed from gene expression data. This graphic illustrates the enormous lower bounds on memory that can be encountered with iterative enumeration algorithms.

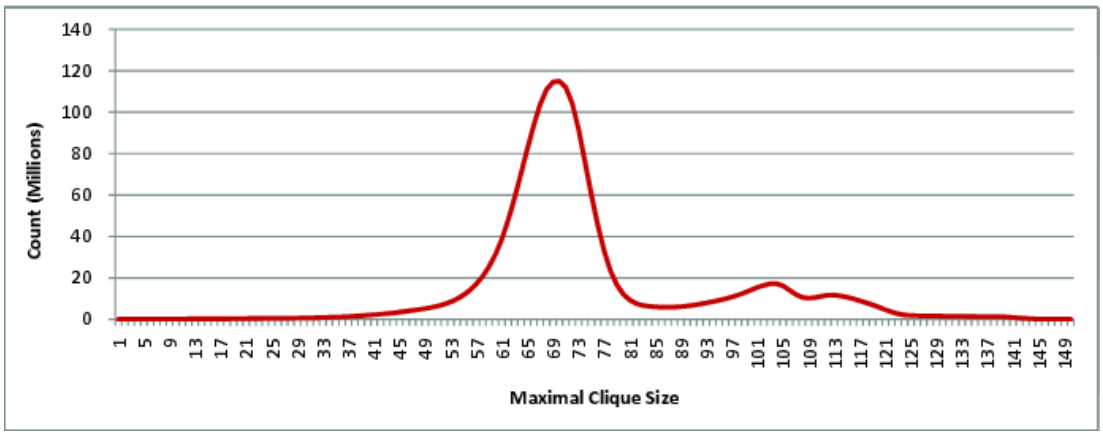

Figure 1. The maximal clique profile of a transcriptomic graph. The graph was constructed from dataset GDS3672 from the Gene Expression Omnibus (GEO) using a correlation threshold of 0.81. Maximal clique enumeration algorithms that are based on a breadth-first traversal of the search tree retain at each step all maximal cliques of a given size. This can lead to titanic memory requirements. This graph, for example, contains more than 110 million maximal cliques of size 70. These sorts of memory demands tend to render non-backtracking methods impractical.

Many variations of backtracking algorithms for maximal clique enumeration have been published in the literature. To the best of our knowledge, all can be traced back to the algorithms of Bron and Kerbosch first presented in [37]. Some subsequent modifications tweak the data structures used. Others change the order in which vertices are traversed. See [38] for a performance comparison between several variations of backtracking algorithms.

But a graph can theoretically contain as many as $3^{n/3}$ maximal cliques [39]. It was shown in [40] that the Bron-Kerbosch algorithm achieves this bound in the

worst case. No algorithm with a theoretically lower asymptotic runtime can thus exist. For a much more complete history and survey of algorithms to find maximum cliques and enumerate maximal cliques, see [34].

1.4.3 The Paraclique Algorithm

Cliques are the most restrictive graph-based cluster, since they require that all possible edges be present. The effects of noise, which is almost inevitably present in experimental data, can generally be overcome by slightly relaxing the extreme stringency of cliques. The paraclique algorithm [19] relaxes the restrictiveness of cliques. The algorithm finds a maximum clique and uses it as a core to build a paraclique. A paraclique consists of the maximum clique and all vertices that are missing no more than g edges to vertices in the maximum clique, where g is called the *glom term*. The paraclique is removed from the graph and saved, and a maximum clique is found in the remaining graph to form the core of the next paraclique. The process continues until a stop condition is reached, typically when the remaining graph contains no clique larger than a user-supplied parameter.

There are several varieties of the paraclique algorithm. One version considers connectivity only to the maximum clique when deciding whether to glom a vertex. Another version considers connectivity to both the maximum clique and to already glommed vertices. Either of these versions can obtain overlap by removing just the maximum clique from the graph with each paraclique found, leaving behind any glommed vertices to potentially be glommed onto later paracliques. In this work, we use only the non-overlapping version that considers connectivity to the maximum clique.

I contributed to the paraclique algorithm by extending the notion of the glom term. Often our analysis produces maximum cliques with several hundred vertices. A small glom term of three or less, as is typically used, may still be too restrictive on very large maximum cliques. Consider, for example, the difference between a clique X of size 5 and another clique Y of size 100. Using a glom term of 3, a vertex connected to 2 vertices of X would be included in a paraclique with X, while a vertex connected to 96 vertices of Y would not be included in a paraclique with Y. But intuitively, in the latter case the vertex seems more significantly associated with vertices in the maximum clique. Therefore I introduced a parameter that scales with clique size, called the *proportional glom factor*, and modified the algorithm accordingly. The proportional glom factor is the minimum proportion of edges that must be present between a vertex v and vertices in the maximum clique in order for v to be included in a paraclique. Research that uses the paraclique algorithm in this dissertation uses a proportional glom factor.

Just as choosing an appropriate threshold to create an unweighted graph is a matter of ongoing research, so too is the selection of an appropriate glom term or proportional glom factor. Figure 2 shows the effect of threshold and proportional glom factor choice on the results of the paraclique algorithm on an mRNA expression dataset. Higher thresholds produce smaller, denser, and fewer paracliques. Furthermore, the higher the proportional glom factor, the smaller, denser, and more numerous the paracliques.

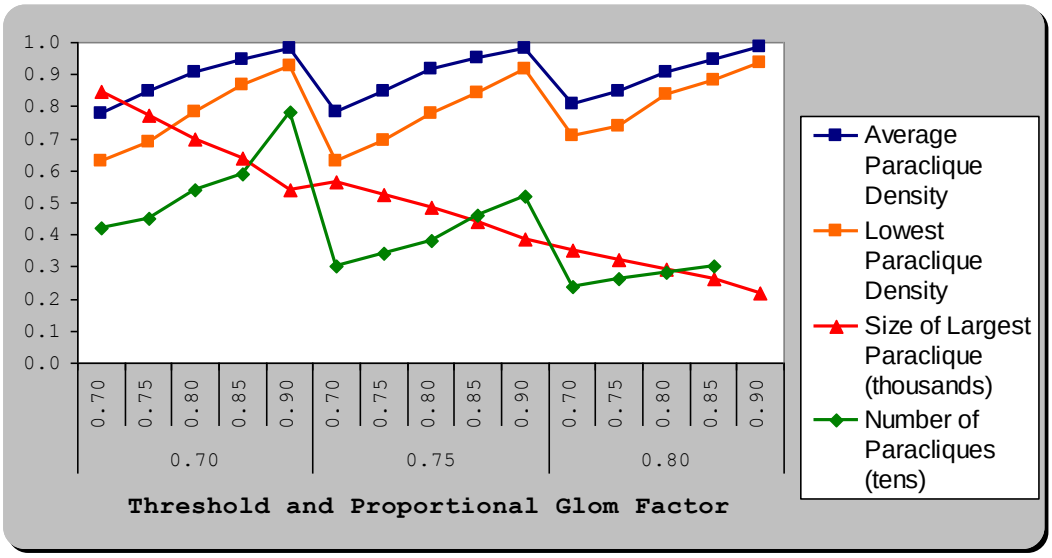

Figure 2. The effect of threshold and proportional glom factor on paracliques. As threshold and proportional glom factor decrease, the average density of paracliques and size of the largest paraclique decreases.

Chapter 2 Algorithms for General Graphs

In this chapter we describe four research topics, related by the common thread of using algorithms for either maximum or maximal cliques in general graphs. The first topic is the search for ethanol responsive gene networks in the prefrontal cortex of mice by analyzing a gene expression dataset. The second uses time series data in the developing mouse cerebellum. The third describes the development and application of a custom clustering algorithm for the prediction of protein-protein interactions. And the fourth is the development, implementation and testing of an algorithm that enumerates all maximum cliques in a graph. Each of the four sections in this chapter consists of previously published material from four publications, edited for this dissertation. The appropriate publication is cited at the beginning of each section, along with a description of my contributions to the work. Wherever possible, I attempt to limit the background to that required to give my contribution context. In the case of the first three sections, 2.1, 2.2 and 2.3, my contribution was as part of the methods section, tailoring our graph-theoretical tools to the analysis of the particular data.

2.1 Ethanol Responsive Gene Networks in the Prefrontal Cortex

Most of this section was published as part of [41] or was presented in preliminary form in the following three posters:

- "Graph Theoretic Analysis of BXD Mouse MRNA Expression Ethanol and Phenotype Data," *33rd Annual Conference of the Research Society on Alcoholism*, San Antonio, TX, June, 2010, C. A. Phillips, A. R. Wolen, M. F. Miles and M. A. Langston.

- "Identifying Ethanol-Regulated Gene Networks in the Mouse Brain using Graph Algorithms," *8th Annual UT-KBRIN Bioinformatics Summit*, Pikeville, TN, March, 2009, C. A. Phillips, A. D. Perkins, A. R. Wolen, E. J. Chesler, M. F. Miles and M. A. Langston.

- "Graph-theoretical Algorithmic Analysis of Microarray Data for Identification of Murine Brain Ethanol-Regulated Gene Networks," *38th Annual Conference of the Society for Neuroscience*, Washington, DC, November, 2008, C. A. Phillips, A. D. Perkins, A. R. Wolen, E. J. Chesler, M. F. Miles and M. A. Langston.

My co-authors designed a microarray experiment on acute ethanol response in the prefrontal cortex of mice. They collected and normalized the microarray data. My contribution included performing eQTL and graph-based analysis on the normalized data and helping to write the paper. The three posters were primarily about my contribution to the research. Only those portions of the paper necessary to provide context for my contributions are included here. I have performed numerous minor edits, both for clarity and to make the style and terminology consistent with the rest of this dissertation, and have reorganized where necessary to integrate sections of the paper and posters into a coherent section.

Individual differences in initial sensitivity to ethanol are strongly related to the heritable risk of alcoholism in humans. To elucidate key molecular networks that modulate ethanol sensitivity, we performed the first systems genetics analysis of ethanol-responsive gene expression in brain regions of the mesocorticolimbic reward circuit (prefrontal cortex, nucleus accumbens and ventral midbrain) across a highly diverse family of 27 isogenic mouse strains (BXD panel) before and after treatment with ethanol.

The impact of acute ethanol on transcript abundance was measured using the Significance-score (S-score) algorithm [42], which utilizes individual probe-level data to determine the statistical significance of transcript level differences between a pair of Affymetrix microarrays. We utilized the R implementation of the S-score algorithm [43] to compare microarray expression levels within BXD strains across treatment groups to generate a saline vs. ethanol S-score for each probeset, where a positive S-score indicates up-regulation with ethanol and vice versa.

S-scores are normally distributed with a mean of 0 and a standard deviation of 1 [42]. For two-tailed tests, p-values for each probeset were calculated as twice the probability of obtaining an S-score at least as large as the absolute value of the observed S-score. Statistical significance of a given probeset's ethanol response across BXD strains was assessed using Fisher's combined probability test [44]. An R implementation of Fisher's method, available as part of the MADAM package [45], was used to combine the S-score transformed p-values. This process was then repeated for 1,000 random permutations of the observed S-score expression matrix, so that empirical p-values could be obtained by comparing observed results to the permutation distribution. Finally, to correct for multiple testing, q-values were generated from the empirical p-values [46]. Probesets with q-values at or below 0.05 were considered to be significantly ethanol-responsive.

2.1.1 Paraclique and Network Analysis

We applied algorithms for both clique-centric clustering and topological analysis to the two murine datasets: one with 43 LXS RI lines and one with 27 BXD RI lines. As described above, each dataset contained a control group, which was administered saline, and a test group, which was given ethanol (1.8 g/kg). For both datasets, Affymetrix M430A 2.0 microarrays were used to measure mRNA expression in the prefrontal cortex (PFC) at four hours post-treatment. Each dataset was normalized using both RMA and S-scores. Shown here are results from the S-score LXS dataset.

Steady-state RMA and saline vs. ethanol S-score expression datasets were analyzed using the paraclique algorithm [19] to identify gene co-expression networks. We first calculated all pairwise Pearson correlations across probesets, where each probeset is represented as a vector of BXD expression values, and used this data to construct an unweighted graph in which vertices represent probesets and edges were present whenever the absolute value of the correlation between two probesets was at least 0.7. The choice of threshold when converting a weighted graph to an unweighted graph is analogous to the choice of p-value when determining significance; it is chosen to produce a reasonable tradeoff between false positives and false negatives. A correlation threshold of |0.7| across 27 strains yields a correlation p-value of 4.8e-05 (calculated using Student's t-distribution). Such low p-values are indicative of the rigor of graph-theoretical techniques. Figure 3 depicts the distribution of correlations and the selected threshold.

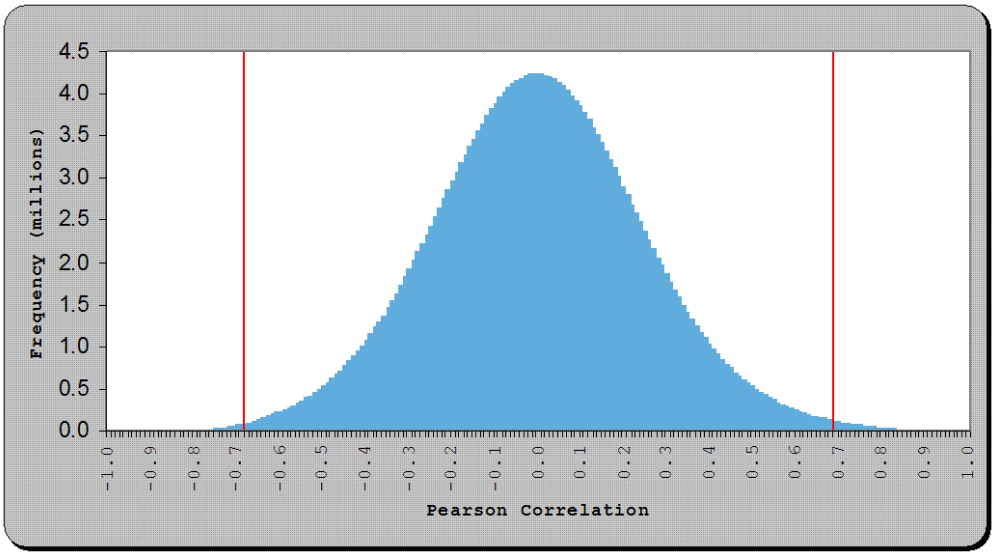

Figure 3. The distribution of Pearson correlations in the BXD S-score dataset. Red lines indicate the threshold of 0.7 chosen to construct the unweighted graph.

Because S-scores are a measure of significance of change in expression of a gene, tightly interconnected groups of vertices in the unweighted graph correspond to groups of genes that have similar differential expression between ethanol and saline across strains. Since genes in such a group all appear to have the same response pattern to ethanol, they form a putative biological network.

Since the inevitable noise in large microarray datasets can render clique too restrictive, we used a relaxed version, the paraclique algorithm. We selected a proportional glom factor of 0.7 for the analyses presented here, which maintains an edge density above 0.9 in nearly all the resulting paracliques. For such defined paracliques, probesets had expression responses to ethanol correlated with at least 70% of the other paraclique members at a threshold at or above |0.7|. Lowering the proportional glom factor below 0.7 resulted in a sharp drop-off in edge density. Furthermore, empirical testing showed that more stringent proportional glom factors produced similar overall functional results but tended to fragment known correlated gene groups (e.g. dopamine signaling genes) into multiple paracliques (data not shown).

The relative importance of each node within a paraclique was assessed using network topological measures of connectivity and centrality. Degree of connectivity was equal to number of edges linking a probeset to other paraclique members, based on the |0.7| edge correlation threshold used to construct the unweighted graphs. Betweenness centrality measures how frequently a node is included in the shortest paths between all pair-wise members of a network. With the edge threshold at |0.7|, Spearman's rank correlations were typically above 0.9 between centrality and connectivity. Increasing the edge correlational threshold to |0.9| reduced the connectivity/centrality correspondence to ~0.6 and greatly increased the centrality for a subset of nodes situated between densely inter-connected subnetworks. We therefore used betweenness centrality scores within unweighted graphs constructed using the more stringent |0.9| edge threshold as a supplemental measure of node importance. Both measures were calculated using the igraph package for R [47]. Fisher's exact test was used to identify paracliques that harbored a greater number of significantly ethanol-responsive probesets than what would be expected by chance. The 30,941 probesets that passed the present-call filter served as the background for this analysis. Paracliques with a Bonferroni adjusted p-value at or below 0.05 were judged to be significantly enriched for ethanol-responsive probesets.

2.1.2 Functional Analysis

Functional enrichment analyses were performed using ToppFun, a functional enrichment application available at toppgene.cchmc.org as part of the ToppGene suite of web applications [48]. Each paraclique was considered on an individual basis. Entrez ID's for all members of a paraclique were submitted and analyzed for over-representation of genes that belong to a Gene Ontology (GO) category (cellular component, molecular function and biological process), biological pathway, gene family or, similarly, encode a particular protein domain. In order to enhance the specificity and informativeness of these results, we considered only those categories that comprise greater than 3 and fewer than 300 genes, inclusive. Multiple testing was accounted for using a 1% FDR threshold. Results were curated by excluding categories with gene lists more than 80% redundant with other, less enriched, categories.

2.1.3 Combining Transcriptomic and Phenotype Data

We used graph-theoretic algorithms in a combined analysis of the BXD S-score data and BXD phenotype data from GeneNetwork's (http://www.genenetwork.org) database of phenotypes. The data consisted of 2137 phenotypes, each with quantitative measurements on up to 92 BXD strains, the BL6 and D2 parental strains, and the two F1 strains.

Combining the 22626 probesets in the Affymetrix microarray with the 2137 phenotypes, we calculated all pairwise Pearson correlations. Since the phenotypes typically did not have measurements for all strains, we translated the Pearson correlation into correlation p-values. We expect the correlations to follow a normal distribution, and indeed that is the case in both the microarray and the phenotype datasets.

Once we calculate all pairwise Pearson correlations, the result is represented by a weighted graph in which each vertex is either a probe or a phenotype and the weight of each edge is the correlation, or in our case the correlation p-value. From the weighted graph we construct an unweighted graph by retaining only those edges with correlation above some threshold t. To maintain similar edge density between probes and phenotypes, we used contrasting values of t for phenotype-phenotype, probeset-phenotype, and probeset-probeset thresholds. See Figure 4. We used a correlation p-value threshold of 0.001 for both probe-probe and phenotype-phenotype edges, and a correlation p-value threshold of 0.01 for probe-phenotype edges since it results in a probe-phenotype edge density between the two intra-type densities.

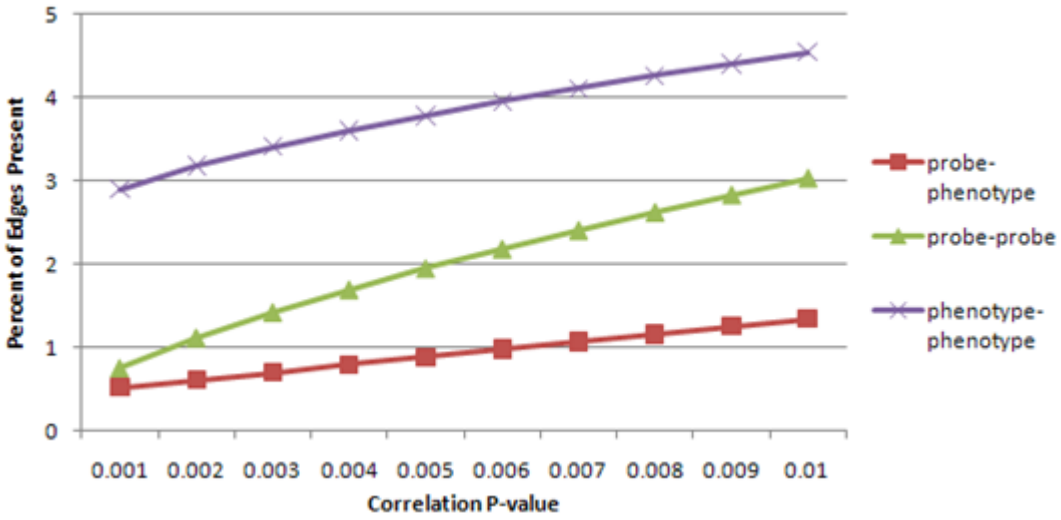

Figure 4. Comparison of edge densities between probes and phenotypes. At the same correlation p-value threshold, the two different data types had different intra-type edge densities. The inter-type density was markedly lower than either inter-type density. At a given threshold for each data type, the inter-type correlation is chosen so that the edge density falls between the intra-type densities.

The analysis resulted in 63 paracliques, from size 293 down to size 12. Many of the paracliques contained only probes or only phenotypes. Four paracliques are of particular interest.

- Paraclique 5 consists of 9 genes and 68 phenotypes. All but 4 of the phenotypes are cocaine response.

- Paraclique 6 consists of 26 genes and 55 phenotypes. All the phenotypes are morphine response.

- Paraclique 26 consists of 5 genes and 21 phenotypes. All the phenotypes are morphine response.

- Paraclique 28 consists of 8 genes 15 phenotypes. All the phenotypes are ethanol response.

The genes in each of these four paracliques form putative networks potentially associated with both ethanol response and the particular phenotype.

2.1.4 QTL Analysis

We used QTL Reaper to perform expression Quantitative Trait Locus (eQTL) mapping for the saline and ethanol treated RMA datasets, as well as the saline vs. ethanol S-score dataset, using a subset of informative microsatellite and SNP markers that have been used to genotype the BXD family [49, 50], and are available from GeneNetwork (genenetwork.org/genotypes/BXD.geno). QTL Reaper is the batch version of WebQTL, available from GeneNetwork. For each gene, we performed interval mapping to find the marker with the maximum Likelihood Ratio Statistic (LRS). A marker with the highest LRS for a gene is the possible location of a QTL.

Genes in a paraclique have a strong tendency to map to relatively few eQTL's on a small number of chromosomes. For instance, 801 of 846 genes in the largest LXS paraclique have loci on only 5 chromosomes (5, 7, 14, 17, and 18). Of these, chromosomes 7 and 18 account for 631 of the 846 gene loci. Furthermore, two eQTL's alone account for 492 genes: rs6394492 on chromosome 7 with 257 genes and rs3720827 on chromosome 18 with 235 genes. Such eQTL enrichment is observed in all 42 paracliques, possibly signaling some ethanol network regulatory function of genes associated with the QTL's.

One of the strengths of combining eQTL mapping with paraclique analysis is the identification of markers that span multiple paracliques. Such markers can aid in identifying genes responsible for trans-network ethanol regulation. The 13 eQTL's in Table 2 appeared in 10 or more of the 42 LXS paracliques.

2.1.5 Maximal Clique Enumeration

Maximal clique enumeration, another graph-theoretic tool, can screen for genes with high network interconnectivity within the graph as a whole. Because of the extent of overlap between maximal cliques, genes that are members of many maximal cliques may be key participants in multiple ethanol-regulated biological networks. The top ten genes by maximal clique count are shown in Table 3.

Table 2. The eQTL's appearing in at least 10 paracliques.

eQTL	*Genes*	*Paracliques*	*Chromosome*
Rs6394492	446	11	7
Rs3720827	395	17	18
mCV24811501	126	15	3
mCV24401139	114	18	3
Rs13483103	88	10	17
D18Mit122	88	10	18
gnf03.037.709	68	12	3
Rs3664070	66	11	3
Rs13477062	66	10	3
Rs4231907	61	12	18
Rs13459176	52	12	15
Rs3707453	35	13	15
Rs3674751	24	13	3

Table 3. The top ten LXS S-score genes by maximal clique count.

Gene	*Maximal Cliques*
C80913	99743705
1110005A03Rik	79324822
Vkorc1	74501989
Timm10	67123030
Gpm6b	65821042
Ndufc1	63138892
Syn2	62637391
5730403B10Rik	54873775
Jtb	53013778
Ptpla	50882906

2.2 Time Series Analysis of the Developing Mouse Cerebellum

Most of this section was previously published in [51]. My co-authors designed and performed a microarray experiment on the developing cerebellum in mice, gathering data at prenatal and postnatal time points. They provided the normalized data, on which I performed graph-based analysis, one of three analyses described in the publication. I also helped write the paper. Only those parts of the paper necessary to describe and provide context for my contributions are included here. As with the previous section, I have made numerous minor edits for clarity and to make the style and terminology consistent with the rest of this dissertation.

There are two major goals in time series analysis. One is to make predictions by extrapolating observed trends into the future. The other is to find temporal patterns in data in order to better understand the processes behind whatever is being measured [52]. When performing time series analysis on gene expression during developmental stages, the first goal, prediction, is of less relevance than the second goal, in the sense that prenatal development occurs but once for an organism. Gene expression during prenatal development may be of limited value at predicting future gene expression in an organism.

The data analyzed in this section deviate from the standard time series model in that the measurements were not taken at evenly spaced intervals. Prenatal measurements were taken at one-day (24-hour) intervals, whereas postnatal measurements were taken at 3-day (72-hour) intervals. The graph-based analysis did not need to specifically account for this deviation.

2.2.1 Data Description

Here we report a novel time series transcriptome database that spans critical embryonic as well as postnatal cerebellar developmental times. We performed microarray analysis on whole cerebellar tissues from two inbred strains of mouse, C57BL/6J (B6) and DBA/2J (D2) at 24-hour intervals in embryonic development from embryonic day 12 (E12) to postnatal day 0 (P0), and at 3-day intervals from P0 to P9 in the postnatal period, and recombinant inbred mice between these two genotypes (BXDR lines) at 3–7 representative time points (E12.5, E15.5, E18.5, P0, P3, P6 and P9), and three mutant lines of mice whose mutant genes are known to target a single cerebellum cell type, the cerebellar granule cell at E15.5 (Math1 KO and meander tail mutant) or at E13.5, E15.5 and E18.5 (Pax6 KO). We illustrate the use of this developmental time series transcriptome data with three bioinformatics analyses: differential equation modeling to predict transcripton and offtime, paraclique analysis to identify genes with similar dynamics, and dynamical system modeling to infer transcriptional causal relationship from the time series data. We also

demonstrate the utility of a new web-based toolkit called Cerebellar Gene Regulation in Time and Space (CbGRiTS) as a data exploration and analysis platform for studying gene regulatory networks in cerebellar development.

A genome-scale microarray analysis was conducted using the Illumina Mouse WG-6v1 Expression Bead Chip platform on pooled RNA samples from microdissected whole cerebellar tissues (3–10) from B6 and D2 strains at 24-hour intervals from E12 to P0 and then every 3 days until P9. The samples were collected from timed-matings to minimize developmental noise. The reliability among biological replicates (typically *N*=3) was examined by calculating the Pearson correlation coefficient between replicates in each group. The median correlation coefficient was 0.99 and the chips with correlation coefficient lower than 0.97 were discarded. After eliminating individual chips with low fidelity, each time point contained 3 biological replicates except the E14 time point of B6 (*N*=2) and P9 of D2 (*N*=1). In the time series data, of the 46,632 probesets in the Illumina platform, 24,257 probes were significantly different (factorial ANOVA, false discovery rate 5%) from the median signal intensity at one or more time points, which indicates that more than half of the transcripts (52%) queried with the microarray platform were dynamically expressed during cerebellar development. In contrast, only 7141 probes showed different temporal expression pattern between the two strains (false discovery rate 5%). These differentially regulated genes are the candidate genes that may underlie strain differences in cerebellar development and function.

To facilitate resource sharing, the CbGRiTS microarray data resource is publicly available at www.cbgrits.org (GEO accession number GSE60437).

As stated, more than half of the genes tested were dynamically regulated during cerebellar development. The developmental time and time window of gene expression would presumably be an indication of a gene's potential role during development.

2.2.2 Paraclique Method

To construct a data structure suitable for graph algorithm clustering, we first calculated all pairwise Pearson correlations between microarray probes across developmental time points. In most analyses, such correlations are expected to follow a reasonably normal probability distribution, and indeed, that is the case for both the B6 and D2 data (Figure 5). Such correlations constitute edge weights in a graph, the vertices being probes. To convert from a weighted to an unweighted graph, we selected a correlation threshold of 0.9 and retained those edges with correlation magnitudes at or above this threshold, discarding edges with weights below the threshold. Selecting an ideal threshold for such graphs has been the

subject of recent study [15-17], but it remains for the most part analogous to the selection of p-value to determine significance. A threshold of 0.9 yields an exceptionally low correlation p-value over the 12 and 13 time points of the B6 and DBA data, which translates into the expectation that dense subgraphs will have very low rates of false positives. Once we obtained an undirected graph, we applied the paraclique algorithm [19] using a proportional glom factor of 0.9 for both strains, meaning that new vertices must be connected to 90% or more of the vertices in the original maximum clique to become part of the resulting paraclique.

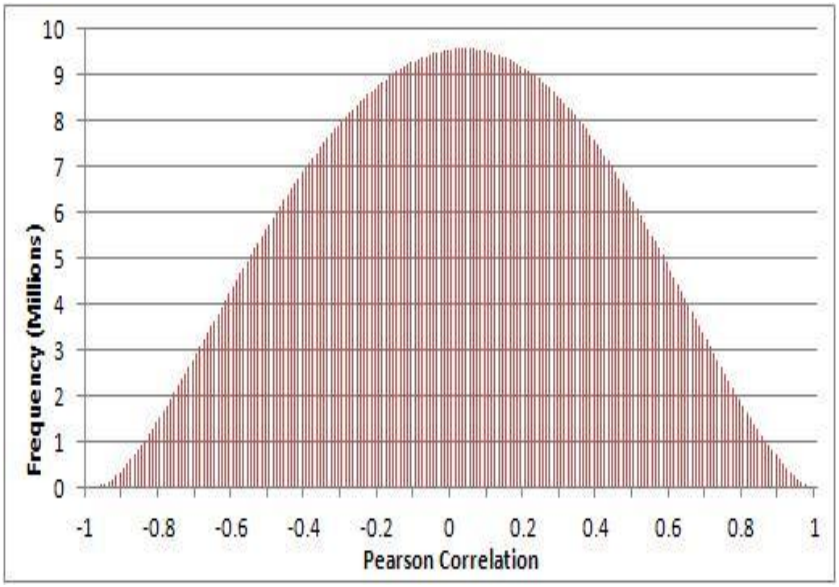

Figure 5. The distribution of Pearson correlations in the B6 strain. For paraclique-based clustering, we calculated all pairwise Pearson correlations between probes in the microarray data across developmental time points. As expected, both the B6 and D2 data showed a reasonably normal probability distribution. Based on the distribution, we used a correlation coefficient threshold of 0.9 to construct the graph and a proportional glom factor of 0.9 to generate paracliques.

2.2.3 Paraclique Results

Next, we explored our data to identify clusters of genes that shared similar expression patterns with the idea that they may be related to common developmental events in the cerebellum. To this end, we employed paraclique analysis to identify sets of genes that share the same temporal expression pattern. The paraclique algorithm clusters transcripts based on correlated temporal expression patterns using graph-based methods [53], with the underlying premise that common temporal expression patterns could be due to common developmental processes and/or transcriptional control mechanisms. Paraclique, *k*-clique

communities and other clique-based clustering algorithms generally tend to produce superior results [21, 31] and are highly resistant to false positives, as compared to commonly used, and less computationally demanding, clustering methods.

Paraclique analysis produced 473 clusters with 12,074 transcripts for the B6 data, and 469 clusters with 10,095 transcripts for the D2 data, using a correlation coefficient threshold of 0.9. The paraclique size ranged from 10 to 381 transcripts (Figure 6). Each paraclique (cluster) represents a group of transcripts whose expression pattern is highly correlated, either positively or negatively, over time. To illustrate this pattern, we generated a "signature" for each paraclique by averaging the expression value of each paraclique element at each time point. Such a signature shows the expression profile across time for the paraclique as a whole (Figure 7). In order to examine the hypothesis that high correlation of temporal expression patterns among paraclique members is due to common developmental processes, we utilized the DAVID Bioinformatics Database [54, 55] to perform Gene Ontology (GO) [56] enrichment analysis. Many of the paracliques showed enrichment for brain and development related categories, indicative of common developmental processes and/or transcriptional control mechanisms. Further, detailed anatomical analysis with in situ hybridization (ISH) databases revealed that many of the genes have expression in the same cell type or in cells derived from the same progenitor pool.

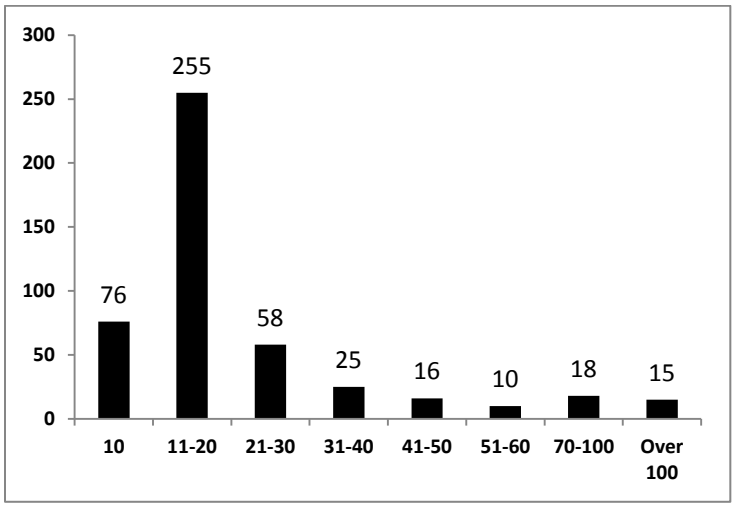

Figure 6. The distribution of the number of genes in paracliques. 70% of paracliques had 20 genes or fewer and only 15 paracliques contained more than 100 genes.

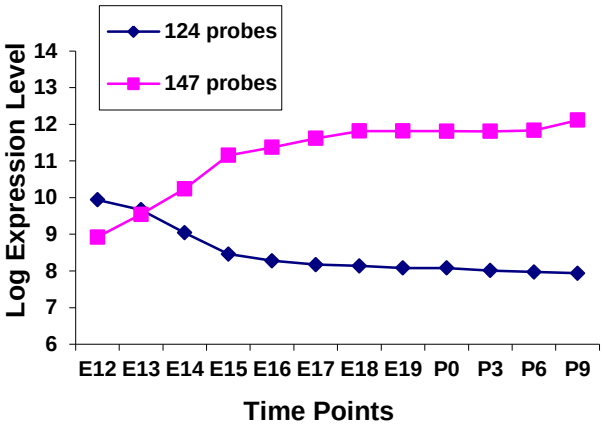

Figure 7. The signature expression pattern of paraclique 3 in the B6 strain. In a paraclique, all genes have high correlation with each other across time. These correlations are both positive and negative. A paraclique can thus be divided into two groups of correlates. Genes in one group have high positive correlation with each other and high negative correlation with genes in the other group. A signature expression pattern is the average expression value of all the members of each group at each time point, and represents the expression profile across time for the cluster as a whole. Because of the negative correlations, the behavior of two groups of genes in a paraclique tends to appear as mirror images of each other. Each paraclique exhibited a distinct expression profile, with different paracliques exhibiting widely varying signatures.

Gene ontology-based enrichment analysis was performed on paracliques containing genes known to be involved in brain development. One example, B6 paraclique 3, had significant enrichment for development-related functional categories. The top 10 most enriched GO terms are listed in Table 4. Paraclique 3 contains 27 members, including Eomes, Plxnb2, Pcsk9 and Lhx9. The GO analysis of the cluster showed enrichment for both brain-related and development-related categories. Although certain portions of the correlation in the temporal domain could be due to general developmental processes, such as cell/tissue growth and proliferation, the highly significant p-values for brain-specific categories indicate that a large portion of paraclique 3 members may have common regulatory mechanisms.

2.3 A Custom Algorithm for Protein-Protein Interaction Prediction

Most of this section was previously published in [57]. My co-authors developed an algorithm to predict interactions between proteins, producing a network (graph) consisting of known and predicted protein-protein interactions. I designed and implemented a custom algorithm to produce overlapping dense subgraphs that met criteria supplied by domain scientists, specifically that there must be a certain number of overlapping subgraphs. Again, only those parts of the paper necessary to

Table 4. The ten most enriched GO categories for B6 paraclique 3.

GO Term	*P-value*
Neuron Differentiation	1.89E-09
Forebrain Development	1.10E-08
Pattern Specification Process	4.49E-08
Neuron Fate Commitment	6.93E-08
Transmission of Nerve Impulse	7.48E-08
Cell Fate Commitment	1.71E-07
Synapse	1.09E-06
Synaptic Transmission	1.35E-06
Regionalization	1.57E-06
Developmental Protein	2.48E-06

describe and provide context for my contributions are included here. And again, I have made numerous minor edits for clarity and to make the style and terminology consistent with the rest of this dissertation.

In this study, we present a comprehensive pairwise analysis and prediction of the entire human PPI network using the principles of short co-occurring polypeptide regions as mediators of PPIs. Through this massive computational analysis, we predict approximately 170,000 PPIs, of which 140,000 have not been reported previously.

Our computationally predicted interactome represents a comprehensive all-to-all interaction network in humans. This network generates a wide range of testable hypotheses concerning biological processes and informs our understanding of the overall architecture of cellular function. Here, we demonstrate the usefulness of this new predicted interactome through prediction of gene functions, experimental verifications and analysis of putative protein complexes.

2.3.1 Motivation

Protein-protein interactions (PPIs) are essential molecular interactions that define the biology of a cell, its development and responses to various stimuli. Physical interactions between proteins can form the basis for protein functions, communications, and regulation and controls within a cell. Such interactions can result in the formation of protein complexes that perform specific tasks. Similarly, internal and external signals are often realized and communicated through the

formation of stable or transient PPIs. Due to their central importance to the integrity of communication networks within a cell, PPIs are thought to involve important targets for drug discovery [58] and are linked to a number of cellular conditions and diseases [59].

Protein interactions can be represented as an interaction network, where the proteins are interactors (nodes) and connections (interactions) are shown as edges. The graph produced by the interaction prediction method consisted of 11194 nodes and 172183 edges. The edges represent both known and predicted interactions. It was expected that between 2000 and 8000 complexes would exist, with significant overlap.

2.3.2 Algorithm

To decompose the predicted protein pairs into putative complexes, we applied a novel algorithm that combines pre-existing graph-theoretic tools with hierarchical clustering concepts. The algorithm has three independent stages: the initialization stage, which consists of generating an initial set of clusters, the merge stage, which determines which two clusters to merge next, if any, and the glom stage, which evaluates vertices for inclusion into a cluster. The initialization stage is run once, after which the merge and glom stages run alternately until either the desired number of clusters is reached or until neither stage results in a change to any cluster.

Since initialization is an independent step, any initial clustering may be used. It is not required that the initial clustering be overlapping, although stages two and three may grow the clusters so that the end result is overlapping. We chose to use the set of all maximal cliques as the initial clustering. The set of maximal cliques forms a natural overlapping clustering of a graph with the most rigid requirements, namely that all edges be present within each cluster. Real-world graphs often have many small and medium sized maximal cliques, and the protein prediction graph is no exception. These clusters are then allowed to merge and grow in stages two and three, gradually relaxing the stringency until the desired number of clusters is reached. To enumerate all maximal cliques, we used the well-known algorithm of Bron and Kerbosch described in [37] with bitwise improvements from [36].

In the merge stage, the overlap of all clusters is evaluated and the two clusters with the highest overlap proportion are merged. If no two clusters overlap by a proportion greater than a parameter m, then no clusters are merged.

In the glom stage, every vertex not already belonging to a particular cluster is considered for inclusion into a cluster in similar fashion to the paraclique algorithm described in [76]. Those vertices with connectivity proportion greater than g, the proportional glom factor, are added to the cluster. The first time through the glom

stage, every cluster is considered. Subsequent glom stages only consider the cluster newly created by the merge stage, as all other clusters have previously been considered. The process is depicted in Figure 8.

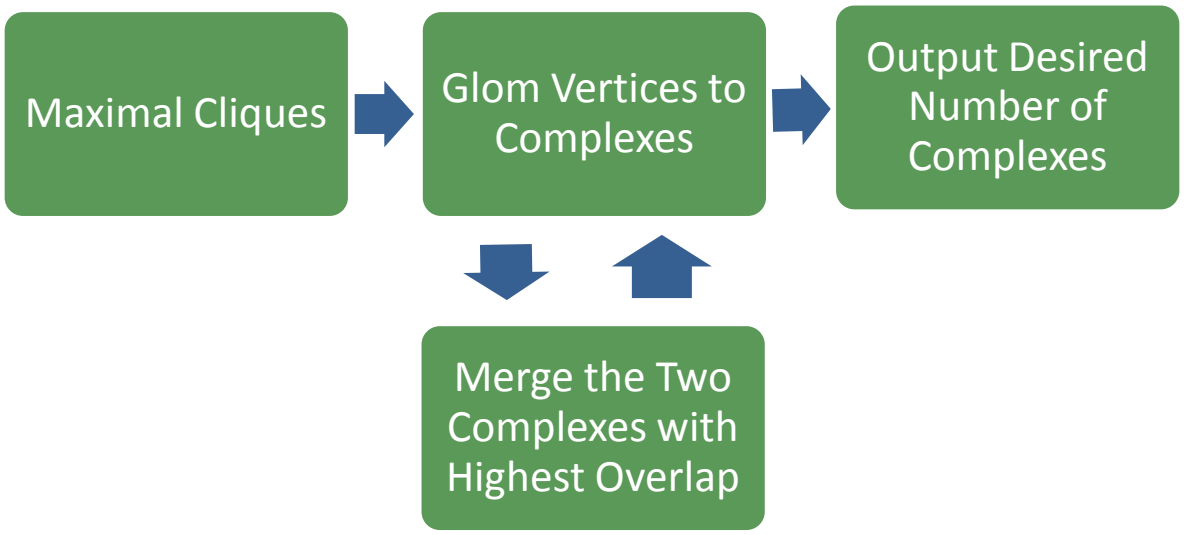

Figure 8. The algorithm to find overlapping protein complexes.

In practice, calculating all pairwise overlaps to find the highest degree of overlap can make the merge stage computationally prohibitive. A small change, however, yields a good approximation version that can be run until the number of clusters is reduced to the point where the exact version can take over. Rather than merging the clusters with the highest overlap, the approximation version merges the first two clusters encountered with overlap at least a, the approximation parameter. For the protein prediction graph, which was initialized with more than 100,000 maximal cliques, we ran the approximation version until the number of clusters reached 20,000, at which point we switched to the exact version. Ultimately, a list of 8,739 paracliques were identified and characterized through a statistical analysis of the GO annotations of each member protein.

2.3.3 Results

Protein complexes can be defined as a group of proteins that interact with each other to form a functional unit. Paracliques [19, 28, 31] can be computationally identified as a sub group of proteins within the interaction network with high degree of interconnectivity and may define putative complexes. Given the size of the human PPI network, prediction of paracliques requires advanced computational approaches

to complete a thorough analysis within a reasonable timeframe. We have applied a novel graph-theoretic approach to automatically identify paracliques within the network (see Methods for details). Our analysis led to a number of interesting predictions. For each paraclique, a statistical analysis of gene ontology (GO) term enrichment was performed. The top GO terms for each paraclique were computed, along with a p-value for the observed enrichment. Here we discuss paracliques 1359, 1409 and 2164 (Figure 9).

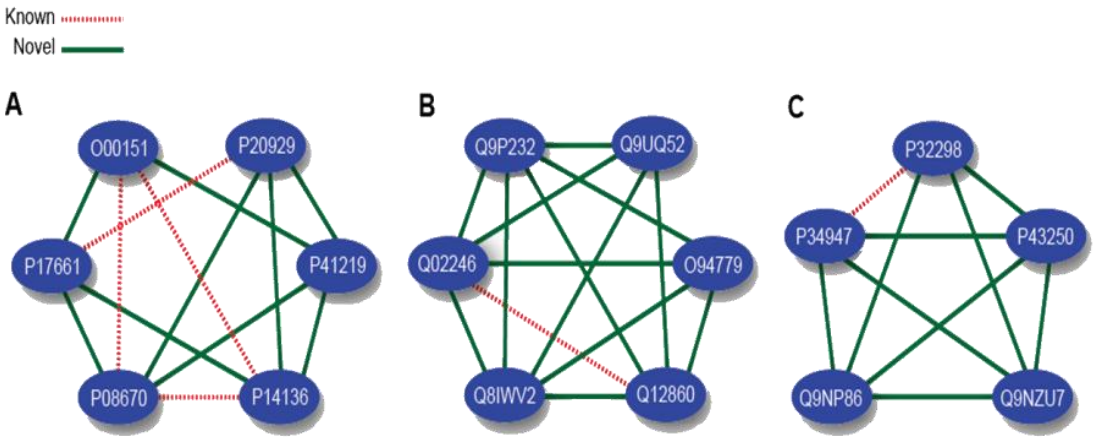

Figure 9. Three protein complexes chosen for further, in-depth analysis. Shown are complex 1359 (A), complex 1409 (B) and complex 2164 (C).

Paraclique 1359 is a complex of six proteins with 13 interactions. O00151 (PDLIM1) is a cytoskeletal protein that acts as an adapter to bridge other proteins (like kinases) to the cytoskeleton. P20929 (NEB) is a muscle protein involved in maintaining the structural integrity of sarcomeres and membranes associated with the myofibrils (F-actin stabilization). The rest of the members (P08670 (VIM), P14136 (GFAP), P17661 (DES) and P41219 (PRPH)) are intermediated filament proteins. On the basis of GO enrichment (p-value 6.5E-07), one may conclude that the activity of this complex is associated with cytoskeleton and structural integrity of the cell.

Paraclique 1409 is a complex of six proteins with 14 interactions. Q02246 (CNTN2) is involved in cell adhesion and the remaining proteins (O94779 (CNTN5), Q02246 (CNTN2), Q12860 (CNTN1), Q8IWV2 (CNTN4), Q9P232 (CNTN3), and Q9UQ52 (CNTN6)) are involved in cell surface interaction during nervous system development. On the basis of GO enrichment, we can assign this complex to cell adhesion (p-value 2.2E-10).

Paraclique 2164 is a complex of five proteins with 10 interactions. Three of its members (P32298 (GRK4), P34947 (GRK5) and P43250 (GRK6)) are G protein-coupled receptor kinase and the remaining two (Q9NP86 (CABP5) and Q9NZU8 (CABP1)) are calcium-binding proteins. Considering the fact that biological interaction between G-protein coupled receptor and calcium-binding proteins has been widely reported and seems essential in signaling pathways, one may conclude that this complex plays a role in G-protein coupled signaling pathway, a claim that is supported by enriched Gene Ontology term (p-value 3.75E-08).

2.4 Maximum Clique Enumeration

Most of this section was previously published in [60]. One of my co-authors designed and implemented an algorithm to enumerate all maximum cliques in a graph. My contribution was to select a suite of publicly-available transcriptomic datasets from which to construct graphs, to help test the algorithm on these graphs, and to help write the paper. I have included most of the paper here, since it is necessary to provide context for my contribution. I have made numerous minor edits for clarity and to make the style and terminology consistent with the rest of this dissertation.

The maximum clique enumeration (MCE) problem asks that we identify all maximum cliques in a finite, simple graph. MCE is closely related to two other well-known and widely-studied problems: the maximum clique optimization problem, which asks us to determine the size of a largest clique, and the maximal clique enumeration problem, which asks that we compile a listing of all maximal cliques. Naturally, these three problems are *NP*-hard, given that they subsume the classic version of the *NP*-complete clique decision problem. MCE can be solved in principle with standard enumeration methods due to Bron, Kerbosch, Kose and others. Unfortunately, these techniques are ill-suited to graphs encountered in our applications. We must solve MCE on instances deeply seeded in data mining and computational biology, where high-throughput data capture often creates graphs of extreme size and density. MCE can also be solved in principle using more modern algorithms based in part on vertex cover and the theory of fixed-parameter tractability. While FPT is an improvement, these algorithms too can fail to scale sufficiently well as the sizes and densities of our datasets grow.

An extensive testbed of benchmark graphs are created using publicly available transcriptomic datasets from the Gene Expression Omnibus (GEO). Empirical testing reveals crucial but latent features of such high-throughput biological data. In turn, it is shown that these features distinguish real data from random data intended to reproduce salient topological features. In particular, with

real data there tends to be an unusually high degree of maximum clique overlap. Armed with this knowledge, novel decomposition strategies are tuned to the data and coupled with the best FPT MCE implementations.

Several algorithmic improvements to MCE are made which progressively decrease the run time on graphs in the testbed. Frequently the final runtime improvement is several orders of magnitude. As a result, instances which were once prohibitively time-consuming to solve are brought into the domain of realistic feasibility.

2.4.1 Background

Any algorithm that relies on maximum clique has the potential for inconsistency. This is because graphs often have more than just one maximum clique. Idiosyncrasies between algorithms, or even among different implementations of the same algorithm, are apt to lead to an arbitrary choice of cliques. This motivates us to find an efficient mechanism to enumerate all maximum cliques in a graph. These can then be examined using a variety of relevant criteria, for example, by the average weight of correlations driven by strain or stimulus [61].

We therefore seek to solve the *Maximum Clique Enumeration (MCE)* problem. Unlike maximal clique enumeration, for which a substantial body of literature exists, very little seems to be known about MCE. The only exception we have found is a game-theoretic approach for locating a predetermined number of largest cliques [62].

While very little prior work seems to have been done on MCE, the problem of maximal clique enumeration has been studied extensively. Since any algorithm that enumerates all maximal cliques also enumerates all maximum cliques, it is reasonable to approach MCE by attempting first to adapt existing maximal clique enumeration algorithms. An implementation of an existing maximal clique enumeration algorithm also provides a useful runtime benchmark that should be improved upon by any new approach. Besides maximal clique enumeration algorithms, another potential strategy is to compute the maximum clique size and then test all possible combinations of vertices of that size for connectivity. While this approach may be reasonable for very small clique sizes, as the maximum clique size increases the runtime quickly becomes prohibitive, and we mention it only for completeness, and focus our efforts on modifying and extending existing algorithms for enumerating maximal cliques.

To enumerate all maximal cliques, we used the well-known backtracking algorithm of Bron and Kerbosch described in [37] with bitwise improvements from [36]. Many variations of backtracking algorithms exist, but as a basis for

improvement, we chose to implement the original algorithm for three reasons. First, an enormous proportion of the time consumed by enumeration algorithms is spent in outputting the maximal cliques that are generated. This output time is a practical limitation on any such approach. Second, since a graph can have as many as $3^{n/3}$ maximal cliques [39], and the Bron-Kerbosch algorithm achieves this bound in the worst case [40], no algorithm with a theoretically lower asymptotic runtime exists. Third, and most importantly, the improvements we introduce do not depend on the particulars of any one backtracking algorithm; they can be used in conjunction with any and all of them.

2.4.2 Results and Discussion

Using the Bron-Kerbosch algorithm as a benchmark, we designed, implemented, and extensively tested three algorithmic improvements, the last based on observations about the nature of graphs produced by transcriptomic data. Along with describing these improvements, we will describe our existing tool for finding a single maximum clique, based on the theory of fixed-parameter tractability [24, 27]. Such a tool is essential for all three improvements, since the first two rely on knowledge of the maximum clique size, and the last uses the maximum clique finding tool as a subroutine. All codes are written in C/C++ and compiled in Linux. For testing, we use 100 graphs derived from 25 different datasets which are publicly available on GEO. We concentrate on transcriptomic data, given its abundance, and eschew synthetic data, having learned long ago that effective algorithms for one have little bearing on the other. (The pathological matchings noted in [63] for vertex cover can be extended to clique, but likewise they too are of course hugely irrelevant to real data.) In an effort to improve performance, we scrutinize the structure of transcriptomic graphs and explore the notion of maximum clique covers and essential vertex sets. Indeed, we find that with the right preprocessing we are able to tailor algorithms to the sorts of data we routinely encounter, and that we can now solve instances previously considered unassailable.

2.4.2.1 Algorithms

In the following sections, we describe each of the MCE algorithms we implemented and tested. The first is the algorithm of Bron and Kerbosch, which we call *Basic Backtracking* and use as a benchmark. Since all our subsequent improvements make use of an algorithm that finds a single maximum clique, we next describe our existing tool, called *Maximum Clique Finder (MCF)*, which does just that. We next modify the Basic Backtracking algorithm to take advantage of the fact that we only want to find the maximum cliques and can quickly compute the maximum clique

size. We call this approach *Intelligent Backtracking*, since it actively returns early from branches that will not lead to a maximum clique. We then modify MCF itself to enumerate all maximum cliques, an approach we call *Parameterized Maximum Clique*, or *Parameterized MC*. In a sense this is another backtracking approach that goes even further to exploit the fact that we only want to find maximum cliques. Finally, based on observations about the properties of biological graphs, we introduce the concepts *maximum clique covers* and *essential vertex sets*, and apply them to significantly improve the runtime of backtracking algorithms.

2.4.2.2 Basic Backtracking

The seminal maximal clique publication of Bron and Kerbosch describes two algorithms. A detailed presentation of the second, which is an improved version of the first, is provided. It is this second, more efficient, method that we implement and test. We shall refer to it here as Basic Backtracking. All maximal cliques are enumerated with a depth-first search tree traversal. The primary data structures employed are three global sets of vertices: COMPSUB, CANDIDATES and NOT. COMPSUB contains the vertices in the current clique, and is initially empty. CANDIDATES contains unexplored vertices that can extend the current clique, and initially contains all vertices in the graph. NOT contains explored vertices that cannot extend the current clique, and is initially empty. Each recursive call performs three steps:

- Select a vertex v in CANDIDATES and move it to COMPSUB.

- Remove all vertices not adjacent to v from both CANDIDATES and NOT. At this point, if both CANDIDATES and NOT are empty, then COMPSUB is a maximal clique. If so, output COMPSUB as a maximal cique and continue the next step. If not, then recursively call the previous step.

- Move v from COMPSUB to NOT.

Note that NOT is used to keep from generating duplicate maximal cliques. The search tree can be pruned by terminating a branch early if some vertex of NOT is connected to all vertices of CANDIDATES.

Vertices are selected in a way that causes this pruning to occur as soon as possible. We omit the details since they are not pertinent to our modifications of the algorithm.

The storage requirements of Basic Backtracking are relatively modest. No information about previous maximal cliques needs to be retained. In the improvements we will test, we focus on speed but also improve memory usage. Thus, such limitations are in no case prohibitive for any of our tested methods. Nevertheless, in some environments, memory utilization can be extreme. We refer the interested reader to [36].

Our Basic Backtracking implementation serves as an initial benchmark upon which we can now try to improve.

2.4.2.3 Finding a Single Maximum Clique

We use the term Maximum Clique Finder (MCF) to denote the software we have implemented and refined for finding a single clique of largest size [30]. MCF employs a suite of preprocessing rules along with a branching strategy that mirrors the well-known FPT approach to vertex cover [27, 64]. It first invokes a simple greedy heuristic to find a reasonably large clique rapidly. This clique is then used for preprocessing, since it puts a lower bound on the maximum clique size. The heuristic works by choosing the highest degree vertex, v, then choosing the highest degree neighbor of v. These two vertices form an initial clique C, which is then iteratively extended by choosing the highest degree vertex adjacent to all of C. On each iteration, any vertex not adjacent to all of C is removed. The process continues until no more vertices exist outside C. Since $|C|$ is a lower bound on the maximum clique size, all vertices with degree less than $|C - 1|$ can be permanently removed from the original graph. Next, all vertices with degree n - 1 are temporarily removed from the graph, but retained in a list since they must be part of any maximum clique. MCF exploits a novel form of color preprocessing [30], used previously in [65] to guide branching. This form of preprocessing attempts to reduce the graph as follows. Given a known lower bound k on the size of the maximum clique, for each vertex v we apply fast greedy coloring to v and its neighbors. If these vertices can be colored with fewer than k colors, then v cannot be part of a maximum clique and is removed from the graph. Once the graph is thus reduced, MCF uses standard recursive branching on vertices, where each branch assumes that the vertex either is or is not in the maximum clique.

2.4.2.4 Intelligent Backtracking

Given the relative effectiveness with which we can find a single maximum clique, it seems logical to consider whether knowledge of that clique's size can be helpful in enumerating all maximum cliques. As it turns out, knowledge of the maximum clique size k leads to a small, straightforward change in the Basic Backtracking

algorithm. Specifically, at each node in the search tree we check if there are fewer than k vertices in the union of COMPSUB and CANDIDATES. If so, that branch cannot lead to a clique of size k, and so we return. See Algorithm 1. While the modification may seem minor, the resultant pruning of the search tree can lead to a substantial reduction in the search space. In addition to this minor change to branching, we apply color preprocessing as previously described to reduce the graph before submitting it to the improved backtracking algorithm. Color preprocessing combined with the minor branching change we call *Intelligent Backtracking*.

```
Algorithm 1: Intelligent Backtracking.
Input: A graph G and the size C of a maximum clique in G
Output: All maximum cliques in G

1  IntBack(COMPSUB, CANDIDATES, NOT)
2      if |COMPSUB| + |CANDIDATES| < C then return
3      if |CANDIDATES| = |NOT| = 0 and |COMPSUB| = C then
4           output COMPSUB, a maximum clique
5      Choose a vertex u ∈ CANDIDATES ∪ NOT
6      For each v ∈ CANDIDATES \ N(u)
7           IntBack(COMPSUB ∪ {v}, CANDIDATES ∩ N(v), NOT ∩
            N(v))
8           CANDIDATES = CANDIDATES \ {v}
9           NOT = NOT ∪ {v}
```

Algorithm 1: Intelligent backtracking. A minor change to the Bron-Kerbosch algorithm uses the precomputed maximum clique size to trim the recursion tree. The input graph has typically been reduced using color preprocessing.

2.4.2.5 Parameterized Enumeration

Given that MCF employs a vertex branching strategy, we investigated whether it could be modified to enumerate not just one, but all maximum cliques. It turns out that MCF, also, lends itself to a straightforward modification that results in enumeration of all maximum cliques. The modification is simply to maintain a global list of all cliques of the largest size found thus far. Whenever a larger maximum clique is found, the list is flushed and refreshed to contain only the new maximum clique. When the search space has been exhausted, the list of maximum cliques is output.

We must take special care, however, to note that certain preprocessing rules used during interleaving are no longer valid. Consider, for example, the removal of

a leaf vertex. The clique analogue is to find a vertex with degree n - 2 and remove its lone non-neighbor. This rule patently assumes that only a single maximum clique is desired, because it ignores any clique depending on the discarded vertex. Therefore this particular preprocessing rule must be omitted once branching has begun.

2.4.2.6 Maximum Clique Covers

If we view MCF as a black box subroutine that can be called repeatedly, it can be used in a simple greedy algorithm for computing a maximal set of disjoint maximum cliques. We merely compute a maximum clique, remove it from the graph, and iterate until the size of a maximum clique decreases. To explore the advantages of computing such a set, we introduce the following notion:

Definition 1. *A* maximum clique cover *of* $G = (V, E)$ *is a set* $V' \subseteq V$ *with the property that each maximum clique of G contains some vertex in the cover*.

The union of all vertices contained in a maximal set of disjoint maximum cliques is of course a maximum clique cover (henceforth MCC), because all maximum cliques must overlap with such a set. This leads to a useful reduction algorithm. Any vertex not adjacent to at least one member of an MCC cannot be in a maximum clique, and can thus be removed.

In practice, we find that applying MCC before the earlier backtracking algorithms yields only marginal improvement. The concept of MCC does, however, lead to a much more powerful approach based on individual vertices. Since any improvement made by MCC is subsumed by the next approach, we do not test MCC by itself.

2.4.2.7 Essential Vertex Sets

Our investigation of the MCC algorithm revealed that it typically does not reduce the size of the graph more than the preprocessing rules already incorporated into MCF. For example, MCF already quickly finds a lower bound on the maximum clique size and removes any vertex with degree lower than this bound. Upon closer examination, however, we found that for 74 of 75 graphs that we initially tested for the conference version of this paper, only one clique was needed in an MCC. That is to say, one maximum clique covered all other maximum cliques. And in our current testbed of 100 graphs, in every case a single maximum clique suffices for an MCC. In fact this coincides closely with our experience, in which we typically see high overlap among large cliques in the transcriptomic graphs we encounter on a regular basis. Based on this observation, we shall now refine the concept of MCC. Rather

than covering maximum cliques with cliques, we cover maximum cliques with individual vertices.

We define an *essential vertex* as one that is contained in every maximum clique. Of course it is possible for a given graph to have no such vertex, even when it contains many overlapping maximum cliques. But empirical testing of large transcriptomic graphs shows that an overwhelming number contain numerous essential vertices. And for purposes of reducing the graph, even one will suffice. An essential vertex has the potential to be extremely helpful, because it allows us to remove all its non-neighbors. We employ the following observation: for any graph G, $\omega(G) > \omega(G/v)$ if and only if v covers all maximum cliques, where $\omega(G)$ is the maximum clique size of G.

We define an *essential set* to be the set of all essential vertices. The Essential Set (ES) algorithm (Algorithm 2) finds all essential vertices in a graph. It then reduces the graph by removing, for each essential vertex, all non-neighbors of that vertex. The ES algorithm can be run in conjunction with any of the backtracking MCE algorithms, or indeed prior to any algorithm that does MCE by any method, since its output is a reduced graph that still contains all maximum cliques from the original graph. As our tests show, the runtime improvement offered by the ES algorithm can be dramatic.

```
Algorithm 2: The Essential Set (ES) Algorithm.
Input: A graph G
Output: A reduced graph G'

1  M = MCF(G), where M is one maximum clique
2  For each v ∈ M
3       G' = G \ v
4       M' = MCF(G')
5       if |M'| < |M| then G = N(v)
6 return G'
```

Algorithm 2: The Essential Set (ES) algorithm.The ES algorithm finds all essential vertices in a graph and removes their non-neighbors.

2.4.2.8 Implementation

We implemented all algorithms in either C or C++. The code was compiled using the GCC 4.4.3 compiler on the Ubuntu Linux version 10.04.2 operating system as well as the GCC 3.3.5 compiler under Debian Linux version 3.1. All timings were conducted in the latter Debian environment on dedicated nodes of a cluster to ensure no effect

on timings from concurrent processes. Each node had a dual-core Intel Xeon processor running at 3.20 GHz and 4 GB of main memory.

2.4.2.9 Testing

In the conference version of this paper, we used three different datasets at 25 thresholds each to derive a total of 75 graphs on which to test our algorithmic improvements. While these graphs certainly sufficed as an initial proof of concept, two concerns could be raised regarding them. First, one might argue that three datasets are not a sufficiently large sample size to provide a true sense of the overall nature of transcriptomic data or an algorithmic improvement's general effectiveness on such data, the large number of thresholds notwithstanding. And second, since the three datasets are proprietary and not publicly available, the results were not as readily reproducible as they might otherwise have been. Obtaining de-identified versions, while feasible, was an unnecessary obstacle to reproducibility.

We address such concerns here by creating a new suite of transcriptomic graphs on which to test our algorithmic improvements. The suite consists of graphs derived from 25 datasets obtained from the Gene Expression Omnibus (GEO) [66], a publicly-accessible repository. For each dataset, graphs were created at four different thresholds, for a total of 100 graphs. The datasets were selected to provide a reasonably diverse sampling of experimental type, species, and mRNA microarray chip type. They cover 8 different species and a number of different experimental conditions such as time series, strain, dose, and patient. Since our graphs are derived from thresholding correlation values, we excluded from consideration any dataset with fewer than 12 conditions. Thresholding correlations calculated using so few conditions can produce unacceptably large rates of false positives and false negatives. The number of conditions range from a low of 12 to a high of 153. Nine of the datasets had not been log-transformed, in which case we performed log-transformation. Four of the datasets contained missing values; in these cases we used correlation p-values rather than correlations for the threshold. Table 5 lists the GEO datasets used for testing.

From the expression data, we first constructed weighted graphs in which vertices represented probes and edge weights were Pearson correlation coefficients computed across experimental conditions. We then converted the weighted graphs into unweighted graphs by retaining only those edges whose weights were at or above some chosen threshold, t. For each dataset, we chose four values for t. All size/density values were within the spectrum typically seen in our work with biological datasets. The smallest graph had 3,828 vertices and 310,380 edges; the largest had 44,563 vertices and 2,052,228 edges.

Table 5. GEO datasets used for testing MCE.

Dataset	*Title*	*Organism*
GDS3505	Seedling roots response to auxin and ethylene availability	Arabidopsis thaliana
GDS3521	Retina response to hypoxia and subsequent reoxygenation:	Mus musculus
GDS3538	Age and diet effect on canine skeletal muscles	Canis lupus familiaris
GDS3561	Occupational benzene exposure: peripheral blood	Homo sapiens
GDS3579	Fer-1 null mutants	Caenorhabditis elegans
GDS3592	Ovarian normal surface epithelia and ovarian cancer	Homo sapiens
GDS3595	Macrophage response to H1N1 and H5N1 influenza viral	Homo sapiens
GDS3603	Renal cancer response to rapamycin analog CCI-779	Homo sapiens
GDS3605	Spared nerve injury model of peripheral neuropathic pain:	Rattus norvegicus
GDS3610	Nasopharyngeal carcinoma	Homo sapiens
GDS3622	Nrf2-deficient lung response to cigarette smoke: dose	Mus musculus
GDS3623	Heart regeneration in zebrafish	Danio rerio
GDS3639	Male and female fruit flies of various wild-type laboratory	Drosophila melanogaster
GDS3640	Copper effect on liver cell line: dose response and time	Homo sapiens
GDS3644	Cerebral palsy: wrist muscles	Homo sapiens
GDS3646	Celiac disease: primary leukocytes	Homo sapiens
GDS3648	Cardiomyocyte response to various types of fatty acids in	Rattus norvegicus
GDS3661	Hypertensive heart failure model	Rattus norvegicus
GDS3672	Hypertension model: aorta	Mus musculus
GDS3690	Atherosclerotic Coronary Artery Disease: circulating	Homo sapiens
GDS3715	Insulin effect on skeletal muscle	Homo sapiens
GDS3716	Breast cancer: histologically normal breast epithelium	Homo sapiens
GDS3703	Addictive drugs effect on brain striatum: time course	Mus musculus
GDS3707	Acute ethanol exposure: time course	Drosophila melanogaster
GDS3692	Lean B6.C-D7Mit353 strain: various tissues	Mus musculus

The number of maximum cliques for the graphs in our testbed ranged from 8 to 74486. As seen with our previous testbed, there was no discernible pattern based on graph size or density. One might ask why there is such wide, unpredictable variability. It turns out that the number of maximum cliques can be extremely sensitive to small changes in the graph. Even the modification of a single edge can have a huge effect. Consider, for example, a graph with a unique maximum clique of size k, along with a host of disjoint cliques of size k - 1. The removal of just one edge from what was the largest clique may now result in many maximum cliques of size k - 1. Edge addition can of course have similar effects. See Figure 10 for an illustrative example.

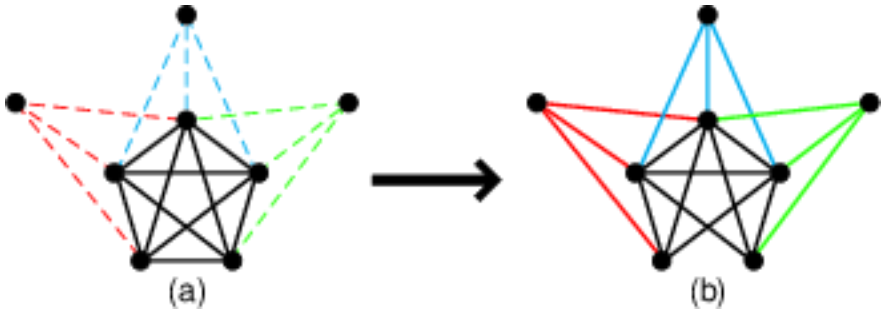

Figure 10. Maximum clique sensitivity. The number of maximum cliques cliques in a graph can be highly subject to perturbations due, for example, to noise. For example, a graph may contain a single maximum clique C representing a putative network of size k, along with any number of vertices connected to k - 2 vertices in C. In (a), there is a single maximum clique of size k = 5, with "many" other vertices (only three are shown) connected to k - 2 = 3 of its nodes. In (b), noise results in the removal of a single edge, creating many maximum cliques now of size k - 1 = 4.

For each algorithm on each graph, we conducted timings on a dedicated node of a cluster to avoid interference from other processes. If the algorithm did not complete within 24 hours, it was halted and the graph was deemed to have not been solved. We chose thresholds to spread the runtimes of the graphs out over the five algorithms we were testing. The largest (smallest in the case of correlation p-value) threshold was selected so that a majority of the algorithms, if not all, solved the graph. The smallest (largest in the case of correlation p-value) threshold was selected so that at least one of the algorithms, but not all, solved the graph.

On each graph we timed the performance of Basic Backtracking, Intelligent Backtracking, and Paramaterized MC. We then reduced the graphs using ES and retested with Intelligent Backtracking and Parameterized MC, in which case the runtimes include both the reduction and the enumeration step. As expected, Basic Backtracking was found to be non-competitive. Both Intelligent Backtracking and

Parameterized MC showed a distinct, often dramatic, improvement over Basic Backtracking. Figure 11 shows the runtimes of each of the five methods on all 100 test graphs. On some of the easier graphs, ones taking less than three minutes to solve, the overhead of ES actually caused a minor increase in the overall runtime. But on the more difficult instances its true benefit became apparent, reducing runtime by an order of magnitude or more. And in all cases where two or fewer algorithms solved the graph, the algorithm was either ES with Intelligent Backtracking, ES with Parameterized MC, or both.

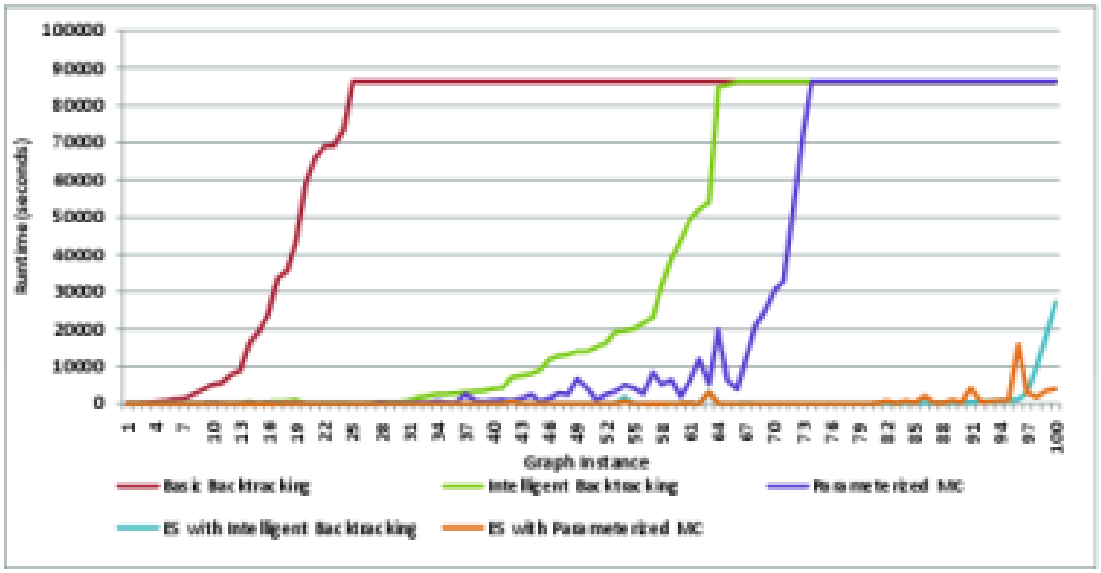

Figure 11. Timings on various approaches to MCE on 100 biological graphs. Timings include all preprocessing, as well as the time to find the maximum clique size, where applicable. Runs were halted after 24 hours and deemed to have not been solved, as represented by those shown to take 86400 seconds. The graph instances are sorted first in order of runtimes for Basic Backtracking, then in order of runtimes for Intelligent Backtracking. This is a reasonable way to visualize the timings, though not perfect, since graphs that are difficult for one method may not be as difficult for another, hence the subsequent timings are not monotonic.

2.4.2.10 Observations

ES serves as a practical example of an innovative algorithm tailored to handle a difficult combinatorial problem by exploiting knowledge of the input space. It succeeds by exploiting properties of the graphs of interest, in this case the overlapping nature of maximum cliques. More broadly, these experiments underscore the importance of considering graph types when testing algorithms.

It may be useful to examine graph size after applying MCC and ES, and compare to both the size of the original graph and the amount of reduction achieved

by color preprocessing alone. Figures 12 and 13 depict original and reduced graph sizes for five graphs we originally tested.

While MCC seems as if it should produce better results, in practice we find it not to be the case for two reasons. First, the vertices in an MCC may collectively be connected to a large portion of the rest of the graph, and so very little reduction in graph size takes place. And second, any reduction in graph size may be redundant with FPT-style preprocessing rules already in place.

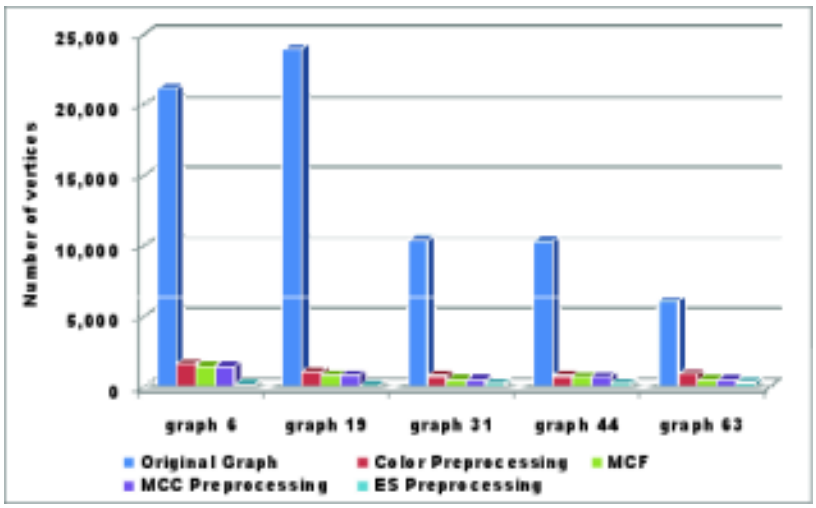

Figure 12. Reduction in graph size for four preprocessing methods. On five representative graphs from our testbed, each of the four preprocessing methods greatly reduces the graph size.

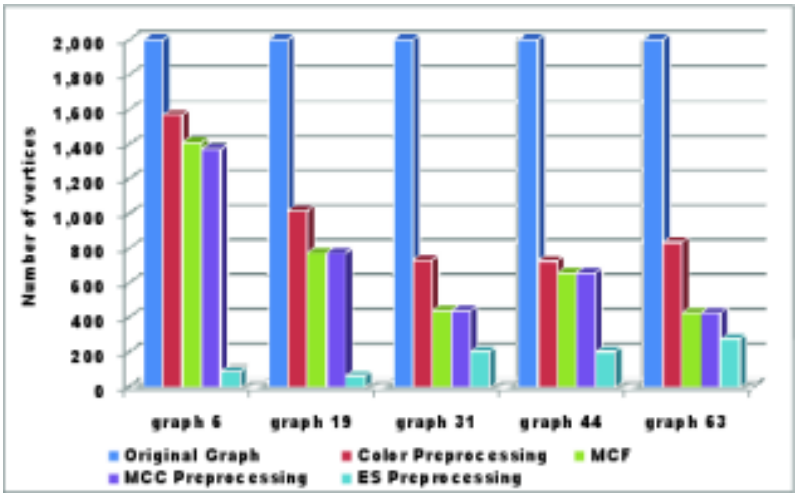

Figure 13. Zoomed view of reduction in graph size from preprocessing. Zooming in on Figure 12 shows how ES preprocessing results in the smallest reduced graph, often leaving only a small fraction of the vertices left by other methods.

It would have probably been fruitless to test and design our algorithms around random graphs. (Yet practitioners do just that with some regularity.) In fact it has long been observed that the topology of graphs derived from real relationships differs drastically from the Erdös-Rényi random graph model introduced in [67]. Attempts to characterize the properties of real data graphs have been made, such as the notion of scale-free graphs, in which the degrees of the vertices follow a power-law distribution [68]. While attempts have been made to develop the scale-free model into a formal mathematical framework [69], there remains no generally accepted formal definition. More importantly, the scale-free model is an inadequate description of real data graphs [70]. We have observed that constructing a graph so the vertices follow a power law (scale-free) degree distribution, but where edges are placed randomly otherwise using the vertex degrees as relative probabilities for edge placement, still results in graphs with numerous small disjoint maximum cliques. For instance, constructing graphs with the same degree distribution as each of the 75 biological graphs in our original testbed resulted in maximum clique sizes no greater than 5 for even the highest density graphs. Compare this to maximum clique sizes that ranged into hundreds of vertices in the corresponding biological graphs. Other metrics have been introduced to attempt to define important properties, such as cluster coefficient and diameter. Collectively, however, such metrics remain inadequate to model fully the types of graphs derived from actual biological data. The notions of maximum clique cover and essential vertices stem from the observation that transcriptomic data graphs tend to have one very large highly-connected region, and most (very often all) of the maximum cliques lie in that space. Furthermore, there tends to be a great amount of overlap between maximum cliques, perhaps as a natural result of gene pleiotropism. Such overlap is key to the runtime improvement achieved by the ES algorithm.

Chapter 3 Algorithms for Bipartite Graphs

This chapter describes algorithms to find dense subgraphs in bipartite graphs and their application to biological data.

3.1 Maximal Biclique Enumeration

Most of this section was previously published in [71]. The description and analysis of the algorithm, along with preliminary performance results, also appears in Yun Zhang's dissertation [72]. Yun designed and implemented an algorithm to enumerate all maximal bicliques in bipartite graphs. My contribution was to improve the speed of the algorithm's implementation and to help test its performance against a recently developed data mining algorithm, LCM-MBC, which achieves the same result. I also performed additional background literature review and helped write the paper. I have taken the liberty of making numerous minor edits for clarity and to make the style and terminology consistent with the rest of this dissertation. Only those parts of the paper necessary to provide sufficient context for my contribution are included here. I therefore omit the description and analysis of the algorithm, which appear in both Yun's dissertation and the published paper.

Integrating and analyzing heterogeneous genome-scale data is a huge algorithmic challenge for modern systems biology. Bipartite graphs can be useful for representing relationships across pairs of disparate data types, with the interpretation of these relationships accomplished through an enumeration of maximal bicliques. Most previously-known techniques are generally ill-suited to this foundational task, because they are relatively inefficient and without effective scaling. In this paper, a powerful new algorithm is described that produces all maximal bicliques in a bipartite graph. Unlike most previous approaches, the new method neither places undue restrictions on its input nor inflates the problem size. Efficiency is achieved through an innovative exploitation of bipartite graph structure, and through computational reductions that rapidly eliminate non-maximal candidates from the search space. An iterative selection of vertices for consideration based on non-decreasing common neighborhood sizes boosts efficiency and leads to more balanced recursion trees. The new technique is implemented and compared to previously published approaches from graph theory and data mining. Formal time and space bounds are derived. Experiments are performed on both random graphs and graphs constructed from functional

genomics data. It is shown that the new method substantially outperforms the best previous alternatives.

The new method is streamlined, efficient, and particularly well-suited to the study of huge and diverse biological data. A robust implementation has been incorporated into GeneWeaver, an online tool for integrating and analyzing functional genomics experiments, available at http://geneweaver.org. The enormous increase in scalability it provides empowers users to study complex and previously unassailable gene-set associations between genes and their biological functions in a hierarchical fashion and on a genome-wide scale. This practical computational resource is adaptable to almost any applications environment in which bipartite graphs can be used to model relationships between pairs of heterogeneous entities.

3.1.1 Background

Bicliques have a long history of applications. The enumeration of maximal bicliques can be traced at least as far back as the seminal work reported in [73]. There the problem was defined in terms of rectangles, binary relations and concept lattices. Subsequent progress on concept lattices was surveyed in [74, 75]. Algorithms for their identification were applied to the analysis of gene co-expression data in [76, 77].

A variety of biological challenges can be addressed by finding maximal bicliques in bipartite graphs. Representative applications include biclustering microarray data [78-80], optimizing phylogenetic tree reconstruction [81], identifying common gene-set associations [19], integrating diverse functional genomics data [82], analyzing proteome-transcriptome relationships [83], and discovering patterns in epidemiological research [84]. Statistical approaches have been applied to some of these problems, but in many cases a discrete approach is beneficial or required because of the structure and diversity of the data under study.

Let us describe a few specific examples. Bicliques have been used in the analysis of gene expression data to represent subsets of genes and subsets of conditions, each pair with a high similarity score [78]. Graph-theoretical approaches have been proposed in this setting to find bicliques in the resultant bipartite graphs that model genes and conditions with vertices, and co-expression levels with edge weights [79, 80, 85]. Bicliques have been used in phylogenetics to improve the accuracy of tree reconstruction [81]. Such a tree denotes evolutionary relationships among species thought to have a common ancestor. Data with no fewer than k genes sampled from no fewer than m species are extracted from sequence databases. This operation is equivalent to finding maximal bicliques with partite set sizes at least k and m. Bicliques have been used in epidemiological research to identify sets

of individuals who share common sets of features. Bipartite graphs can help capture relationships between organisms and a wide range of factors. Maximal bicliques are particularly useful in case-control studies involving categorical features such as genotypes and exposures [84].

Our work has been largely motivated by the computational demands of systems like GeneWeaver [82, 86], a web-based software platform for the integration of functional genomics data. GeneWeaver includes a database containing lists of genes from diverse sources, along with descriptive metadata associated with these lists. Through gene homology, the lists can be combined across species such that genes on the lists are translated to a common reference. This enables the construction of a bipartite graph, with vertices representing individual genes. A suite of tools built on the enumeration of maximal bicliques and other bipartite analyses allows the user to identify groups of genes that are associated with related biological functions, all without any prior knowledge or assumption about such group associations. Efficiency and scalability are paramount, because real-time maximal biclique enumeration is required for web-based user-driven analyses, as well as for effective computations over the entire data repository.

3.1.1.1 The Maximal Biclique Enumeration Problem

In each of the aforementioned applications involving an integration of multiple sets of genome-scale data, bipartite graphs can be used to represent relationships across pairs of heterogeneous data types. An interpretation of such a relationship is accomplished through an enumeration of maximal bicliques. Let us be precise about what this means. A bipartite graph is one whose vertices can be partitioned into a pair of non-empty, disjoint partite sets such that no two vertices within the same partite set are connected by an edge. Let *G* denote a bipartite graph, let *U* and *V* denote its two partite sets, and let *E* denote its edge set. A biclique in such a graph is a complete bipartite subgraph, that is, a bipartite subgraph containing all permissible edges. The notion is formalized as follows:

Definition 2. *Let* $G = (U \cup V, E)$ *denote a bipartite graph. A biclique* $C = (U', V')$ *is a subgraph of* G *induced by a pair of two disjoint subsets,* $U' \subseteq U$ *and* $V' \subseteq V$*, such that* $\forall u \in U', v \in V', (u, v) \in E$.

A *maximum biclique* is a largest biclique in a graph. Unlike the well-known maximum clique problem, there are two distinct variants of the maximum biclique problem: the *vertex maximum biclique* problem and the *edge maximum biclique* problem. The former asks that we find a biclique with the largest number of vertices,

and can be solved in polynomial time [26]. The latter asks that we find a biclique with the largest number of edges, and is $\mathcal{NP}$-complete [87]. In biological applications, the edge maximum biclique is often desirable because it models more balanced connectivity between the two vertex classes. For example, an edge maximum biclique may group together numerous related biological processes and a modest set of their common genes, whereas a vertex maximum biclique may instead group together only a tiny set of related biological processes with great numbers of common genes.

A *maximal biclique* is one not contained in any larger biclique. Examples of maximum and maximal bicliques are shown in Figure 14. The enumeration version of our problem is to find all maximal bicliques in a bipartite graph. In so doing, it turns out that we actually generate both edge maximum and vertex maximum bicliques. Thus, we are chiefly concerned with this enumeration problem, formalized as follows:

Input : A bipartite graph $G = U \cup V, E)$.

Output: All maximal bicliques, or subsets U' of U and V' of V, for which the induced subgraph $G[U' \cup V']$ is complete, and there are no subsets $U'' \supseteq V'$, or $U'' \supseteq U'$ and $V'' \supseteq V'$, such that $G[U'' \cup V'']$ is also complete.

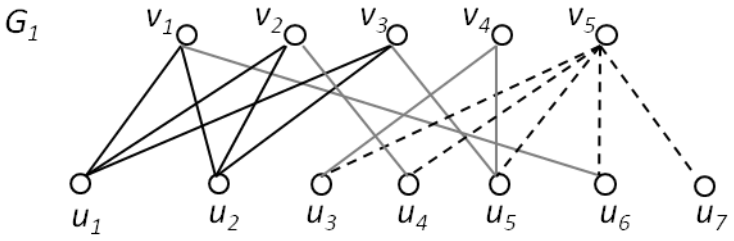

Figure 14. Maximum and maximal bicliques. A bipartite graph G_1 has an edge maximum biclique $B_1(\{u_1,u_2\},\{v_1,v_2,v_3\})$ with 5 vertices and 6 edges, and a vertex maximum biclique $B_2(\{u_3,u_4,u_5,u_6,u_7\},\{v_5\})$ with 6 vertices and 5 edges. Both B_1 and B_2 are maximal.

As observed in [88], the maximal biclique enumeration problem cannot be solved in polynomial time since the number of maximal bicliques may be exponential in the graph size. Nevertheless, there remains a demand for efficiency, because we often need exact solutions to large-scale instances in real time. The Maximal Biclique Enumeration Algorithm (MBEA) that we will define here finds all maximal

bicliques. It exploits structure inherent in bipartite graphs. It employs a branch-and-bound technique to prune non-maximal candidates from the search tree. Its pruning is accelerated by directly removing dominated vertices from the candidate set. Our experimental results demonstrate that the resultant reduction in search space enables MBEA to scale to the tens of thousands of nodes currently encountered in analyzing large biological data sets. In addition, we created an improved version, iMBEA, that selects candidate vertices in the order of common neighborhood size and that uses an enhanced version of branch pruning.

3.1.1.2 Related Work

With widespread applications such as those just discussed, one would expect a plethora of algorithms targeting maximal bicliques on bipartite graphs. Most algorithms that achieve this purpose, however, are either not tailored for bipartite graphs or not designed specifically for maximal biclique enumerations. Most existing graph algorithms for solving this problem fall into two main categories: (i) those designed for bipartite graphs but that either place undue restrictions on the input or require reduction to other problems, and (ii) those designed for general graphs and are thus unable to take advantage of bipartite graph structure. See [72] for a list of these algorithms, their inputs and outputs (with restrictions, if any), and the methods they use.

3.1.1.3 Algorithms for Bipartite Graphs

Existing algorithms for finding maximal bicliques in bipartite graphs are further divided into the following three approaches: exhaustive search with restrictions on outputs, reduction to the clique enumeration problem on general graphs, and reduction to the frequent itemset mining problem in transaction databases.

The most intuitive approach entails exhaustively building all subsets of one partite set, finding their intersections in the other partite set, and checking each for maximality. Algorithms based on exhaustive search must generally place one or more restrictions on the problem to reduce its enormous search space. Moreover, exhaustive search requires storing generated bicliques to determine their maximality. An iterative algorithm is presented in [81] to build subsets progressively, from pairs of vertices to collections of larger and larger sizes. It limits the sizes of both biclique partite sets, yet still requires enormous amounts memory to store the lists used to generate subgraphs and decide maximality. The algorithm described in [84] builds bicliques based on set expansion and extension operations. It employs a hash table that determines maximality to avoid pairwise biclique comparisons, and a queue to maintain bicliques prioritized by figure-of-merit values

(e.g., p-values). Users can specify constraints on the figure-of-merit values to filter out bicliques of insufficient interest.

The second approach relies on graph inflation. As observed in [89], the enumeration of maximal bicliques in a bipartite graph can be transformed into the enumeration of maximal cliques in a general graph by adding all possible edges between vertices within the same partite set, thereby transforming each of the two disjoint vertex sets into a clique. Intuitively, this approach seems neither practical nor scalable. The enormous number of edges that may be needed would seem to result in a concomitant increase in problem difficulty. Given a bipartite graph $G = (U \cup V, E)$ where $|U| = m$, $|V| = n$, $|E| = e$, the number of edges needed to transform G to a corresponding graph G' is $\binom{n}{2} + \binom{m}{2}$. Thus, this method transforms the problem of finding maximal bicliques in a bipartite graph with edge density $\frac{e}{mn}$ to the problem of finding maximal cliques in a graph G' with density $d(G') = \frac{e+\binom{n}{2}+\binom{m}{2}}{\binom{m+n}{2}}$. Note that G' might be dense even if G is sparse. When G has two vertex sets of equal size and no edges (i.e. $|U| = |V| = n$, $|E| = 0$, G' has a density $\frac{n^2}{2n^2-n} \simeq 50\%$. But as we shall see in chapter 4, the intuitive notion that adding so many edges results in a more difficult problem instance does not necessarily hold in practice.

A third approach comes from the field of data mining. It was observed in [90] that a transactional database can be represented by a bipartite graph, with a one-to-one correspondence between frequent closed itemsets and maximal bicliques. A subset of items is defined as a *frequent itemset* if it occurs in at least *s* transactions, where *s* is a parameter called the *support*. On one hand, a frequent itemset and the set of transactions containing the frequent itemset form a biclique. On the other hand, the adjacency lists of a bipartite graph can be viewed as a transaction database by treating each vertex in one partite set as an item and each vertex in the other partite set as a transaction that contains a subset of items. A biclique can thus be mapped to a frequent itemset. A maximal biclique corresponds to a frequent closed itemset, where a frequent itemset *I* is said to be *closed* if the set of transactions containing *I* do not contain a superset of *I*. The *support* of a frequent itemset is the number of transactions in which the set occurs. Enumerating all maximal bicliques is equivalent to enumerating all frequent closed itemsets with support at least 1. Figure 15 shows a mapping between these two problems. A correspondence between maximal bicliques of a general graph and frequent closed itemsets has been shown [91], leading to the suggestion that FPclose and similar frequent itemset mining methods [92-96] may be helpful in enumerating maximal bicliques. Implementations of this approach require a post-processing step to obtain the

transaction set for each frequent closed itemset, as described in [97]. This is because the published methods output only the frequent itemsets (which correspond to half bicliques). Although this post-processing step is straightforward enough, it can be prohibitively time-consuming when the number of maximal bicliques is large. Moreover, known methods take the support level as an input parameter, and find only frequent closed itemsets above the given support. (In general, the lower the support, the longer the algorithms take. A support of 1 is the most difficult, since at this level all frequent closed itemsets must be found.)

Transaction	Items
1	ABDEF
2	ABF
3	BCDE
4	ABCE
5	CDE
6	CDEF
7	AEF

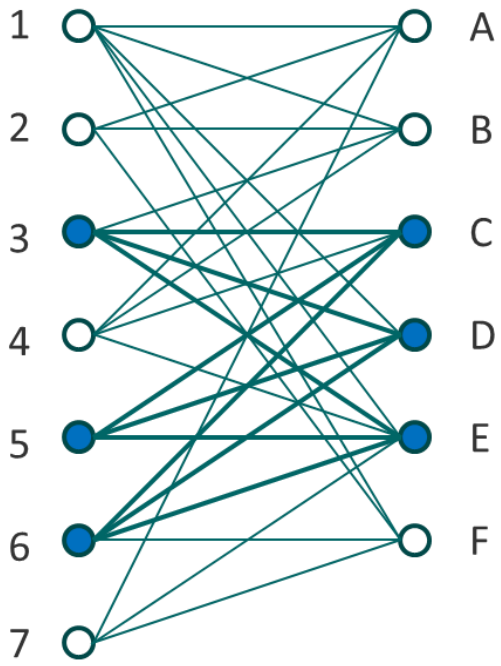

Figure 15. Equivalence between closed itemsets and maximal bicliques. A closed itemset is a set of items for which no superset of items appear in the same transactions. Each closed itemset in a set of transactions corresponds to a maximal biclique when the transactions are depicted as a bipartite graph. The closed itemset {C,D,E} corresponds to the maximal biclique {C,D,E,3,5,6}. Other closed itemsets/maximal bicliques include {A,E,1,4,7}, {A,E,F,1,7}, {B,C,E,3,4}, {B,D,E,1,3}, {A,B,D,E,F,1} and several others.

3.1.1.4 Algorithms for General Graphs

Maximal bicliques can also be found with algorithms designed for general graphs. Such algorithms of course lack any efficiency gains that might be accrued from utilizing bipartite graph structure. The maximal biclique enumeration problem was studied from a theoretical viewpoint in [88], where the focus was on graphs of bounded arboricity. It was proved that all maximal bicliques in a graph of

order n and arboricity a can be enumerated in $O(a^3 2^{2a} n)$ time. This approach is not practical for large graphs, however, because it is unrealistic to expect that arboricity would be limited in practice [90]. A suite of consensus algorithms was presented in [98] for finding complete bipartite (but not necessarily induced) subgraphs. Unfortunately, these algorithms need to keep all maximal bicliques in memory. The Modular Input Consensus Algorithm (MICA), the most efficient among them, has space complexity $O(B)$ and time complexity $O(n^3 B)$, where B denotes the number of maximal bicliques. An algorithm (MineLMBC) based on divide-and-conquer was proposed in [99] to mine large maximal bicliques from general graphs by putting size constraints on both vertex sets to iteratively prune the search space. The algorithm reduces the space complexity to $O(n^2)$ and the time complexity to $O(n^2 B)$. The algorithm on dense graphs from the 2nd DIMACS Challenge benchmarks outperforms MICA when minimum biclique sizes are constrained by certain thresholds.

To solve the biclique enumeration problem, restrictions on either inputs or outputs have been proposed to reduce the search space. These include bounding the maximum input degree [79], bounding an input's arboricity [88] and bounding the minimum biclique size [81, 99] or figure-of-merit [84]. Naturally, no algorithm relying on these restrictions can solve arbitrary bipartite instances.

3.1.2 Implementation and Testing

We implemented MBEA and iMBEA and compared them to existing implementations of what should be the two strongest competitors: MICA [98], currently the fastest graph-theoretical algorithm for finding bicliques in general graphs, and LCM-MBC [97], currently among the most advanced data mining algorithms for finding pairs of frequent closed patterns, improving upon LCM [96]. An efficient implementation of MICA is available at http://genome.cs.iastate.edu/supertree/download/biclique/README.html. Efficient codes for LCM can be found at http://fimi.ua.ac.be/src/. Version 2 is reported to be the faster of the two available LCM implementations. The authors of [97] graciously provided us with their implementation of LCM-MBC, which we used in our comparisons. MBEA/iMBEA and MICA accept graphs in a simplified DIMACS edge list format. LCM/LCM-MBC is not DIMACS compatible, however, and required us to convert an edge list into an equivalent adjacency list for the smaller partite set. Graphs come in many formats, of course, so we did not charge any time for this simple conversion.

All implementations were compiled on and timings performed under the Ubuntu 12.04 (Precise Pangolin) x64 operating system on a Dell OptiPlex 9010

Minitower with an Intel Core i7-3770 3.4 GHz processor, 16.0 GB DDR3 non-ECC SDRAM memory at 1600 MHz (4 DIMMs), and a 500 GB 7200 RPM SATA hard drive. Only sequential implementations of MBEA, MICA and LCM-MBC were compared, each making use of a single compute core. MBEA and iMBEA were written in C and compiled with the GNU gcc compiler with O3 optimization turned on. The MICA and LCM-MBC implementations were also complied with the -O3 flag. The wallclock running times we report include both I/O and computation, but exclude the time taken to print out the maximal bicliques. They are the average of ten, five or three runs for graphs that can be finished within one minute, one hour or three days, respectively. Runs that exceeded three days were killed and omitted from the averages. We employed standard data reduction techniques to reduce the size of bipartite graphs for all methods tested. For example, during pre-processing, two or more vertices with the same neighborhood are merged into a single vertex; this process is reversed at post-processing.

3.1.2.1 Biological Graphs

We tested the algorithms on biological graphs derived from functional genomics data. One set of graphs, which was extracted from cerebellum data, was created using a matrix of correlation p-values for gene expression to phenotypes across strains of mice in a single population [100]. The matrix consists of 45137 genes represented by microarray measures of transcript abundance and 782 phenotypes to which the transcript abundances are correlated. A bipartite graph is obtained by placing an edge only where the correlation p-value is at or below some preset threshold. The density of this graph can be varied by adjusting the threshold. The lower the p-value threshold, the lower the graph density. To test a wide variety of densities, we created twenty graphs over a range of thresholds, from 0.01 to 0.20, with a step of 0.01.

The second set of graphs, which represent phenotype-gene associations, was created from a correlation matrix between 33 phenotypes and 17539 genes, calculated over a panel of more than 300 mice. For each threshold, a phenotype-gene edge is present if the correlation is at or above the threshold. We created graphs with a range of thresholds, so that the lowest threshold ran in a small fraction of a second and the largest in tens of minutes.

In both sets, edge density increases across the range of thresholds, from roughly 0.2% to about 2.5% in the cerebellum graphs, and from roughly 6.6% to as high as 37.4% in the pheno-gene graphs. Computational demands increase even more rapidly, because the number of maximal bicliques tends to grow exponentially with a linear increase in threshold values.

3.1.2.2 Random Graphs

In addition to biological graphs, we tested iMBEA and LCM-MBC on random bipartite graphs, using two different random graph models. The first is the classic Erdős-Rényi random graph model. Here, we fixed the number of vertices in each partite set at 300 and varied the density from 0.1 to 0.28. The density range was selected so that the lowest would run in well under a second and the highest would require several minutes. We also tested graphs with 400 and 500 vertices, but the results were similar enough to graphs with 300 vertices that we omit their discussion.

For the second random graph model, we modified the Erdős-Rényi model so that we could study graphs with both high and low degree variability. The graph generator takes as input these four parameters: the size m of the larger partite set, the size n of the smaller partite set, the average vertex degree μ in the smaller partite set, and the coefficient of variation CV of the degrees in the smaller partite set. (Recall that $CV = \sigma/\mu$, where σ is the standard deviation and μ is the mean.) These specifications were used to assign vertex degrees to the smaller partite set. No edges were produced within a partite set, of course. The assigned degrees in the smaller partite set were used to place edges, selecting each endpoint in the larger partite set with uniform probability. For example, if a vertex in the smaller partite set had been assigned degree three, then three neighbors for it were uniformly selected from the larger partite set.

We created three sets of random graphs with this graph generator. The first set fixed the number of vertices in one partite set at 10,000 and in the other partite set at 1000, the edge density at 4.5%, and varied the CV from 0.3 to 1.2. The purpose of this set was to test the behavior of MBEA versus iMBEA when the CV is varied, it being our intuition that iMBEA might be better suited to graphs with higher CV. The second and third sets of graphs were created to test iMBEA versus LCM-MBC when the relative partite set sizes were varied. In one set, the size of the larger partite set is fixed at 10,000 and the size of the smaller partite set is varied from 100 to 1000. In the other set, the size of the smaller partite set is fixed at 500 and the size of the larger partite set is varied from 5000 to 50,000. In both sets we used an edge density of 3.0%, which provided a wide spectrum of partite set sizes while keeping runtimes within reason.

3.1.3 Results and Discussion

In this section, we compare runtimes of the various algorithms. MICA turns out not to be competitive on any of our graphs. We therefore exclude its timings from our presentation. For instance, iMBEA outperforms MICA by more than three orders of

magnitude on even modest-sized biological graphs. On a somewhat larger graph, iMBEA finishes in under an hour while MICA runs for over three days without completion. And on the largest graphs, MICA runs out of memory. Thus, we feel it is manifest that MICA does not belong in the same class as algorithms such as MBEA and iMBEA, which are specifically targeted at bipartite graphs. We first concentrate on MBEA and iMBEA on both biological and random graphs in order to demonstrate the performance gained by iMBEA's improved pruning. We then move on to compare iMBEA and LCM-MBC on two sets of biological graphs and three sets of random graphs.

3.1.3.1 Comparison of MBEA and iMBEA

In Figure 16 we compare the runtimes of MBEA and iMBEA on the twenty cerebellum graphs. The curves cross at a p-value threshold of about 0.07. iMBEA is roughly three times as fast as MBEA at around threshold 0.20. These results confirm our expectations that the relative simplicity of MBEA wins on sparse graphs produced at lower thresholds, while the improvement overhead of iMBEA more than pays for itself once higher thresholds generate graphs that are sufficiently dense. We also compared MBEA and iMBEA on random bipartite graphs. As shown in Figure 17, while reasonably close, iMBEA consistently outperforms MBEA. The sorted candidate vertex selection and enhanced pruning of iMBEA appear still to produce performance gains. These gains are not as significant, however, as they were for biological graphs. This may be due at least in part to the rather smoothed overall topology of random graphs, as opposed to the uneven density and highly irregular features typically seen in graphs like those in GeneWeaver. To look closer into this behavior, we varied the *CV* with which random graphs were built. We found, as illustrated in Figure 18, that iMBEA outperforms MBEA on random bipartite graphs over the entire *CV* range tested. The performance gap is smaller when the *CV* is low, probably due to MBEA's relative simplicity and reduced overhead. As the *CV* increases, however, the performance gap between MBEA and iMBEA widens. These results help explain iMBEA's superior performance on biologically-derived graphs, which very often exhibit high variation in vertex degree. When comparing our algorithms to other methods, we employ only iMBEA for simplicity. It is possible that on some inputs MBEA would do slightly better.

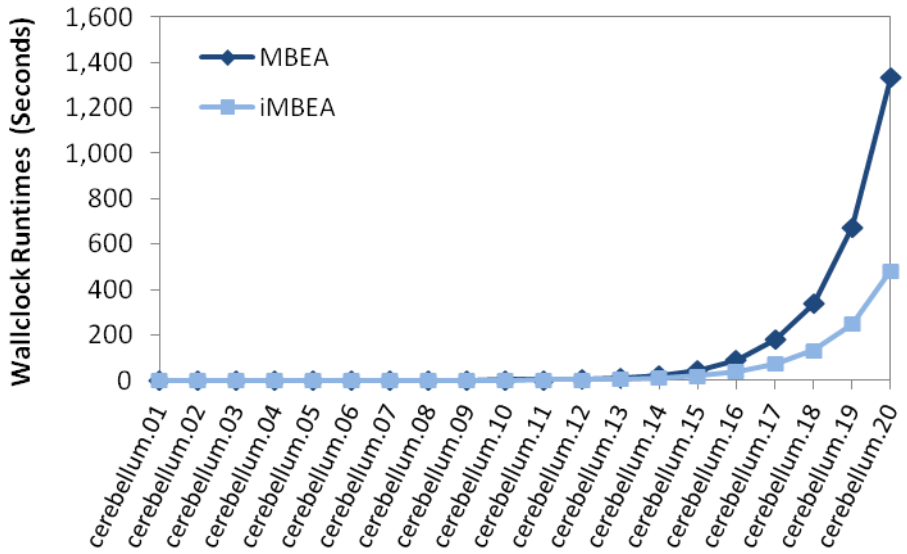

Figure 16. Performance comparison of MBEA and iMBEA on 20 cerebellum graphs from GeneWeaver. As the size and density of the graphs increases, the small overhead incurred by iMBEA's pruning checks is quickly rewarded with performance gains from the additional pruning.

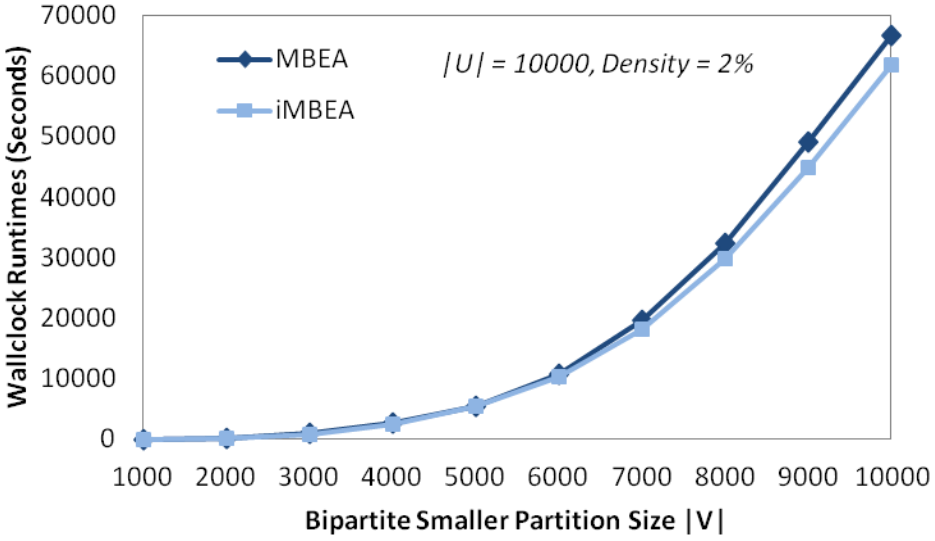

Figure 17. Performance of MBEA versus iMBEA on random graphs. Although runtimes are close, iMBEA consistently outperforms MBEA on random graphs.

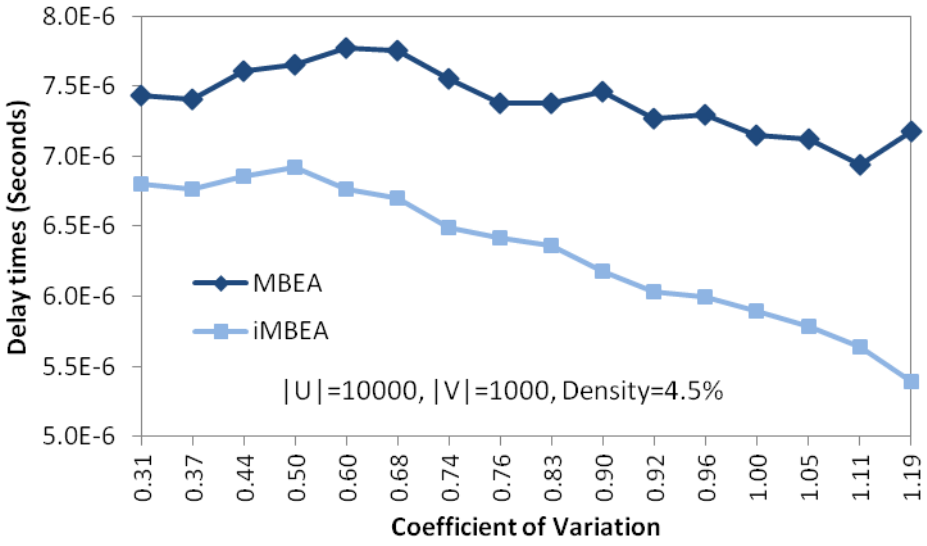

Figure 18. Effect of graph degree structure on MBEA and iMBEA. The average delay time of MBEA and iMBEA on random graphs with the same size and density, but varying degree distribution. On graphs with low coefficient of variation, the performance gap between MBEA and iMBEA is narrower than on graphs with high coefficient of variation. This confirms our expectation that the pruning enhancements of iMBEA have a larger effect on graphs with diverse degree structure.

3.1.3.2 Comparison of iMBEA and LCM-MBC

Figure 19 shows the average runtimes of iMBEA and LCM-MBC on the biological graphs tested. The top chart shows the phenotype-gene graphs, and the bottom two charts show two ranges of p-values for the cerebellum graphs. The performance disparity is most notable when the graphs grow denser. On both the cerebellum and pheno-gene graphs, the maximal bicliques in the densest graph exceed the 2 GB disk storage limit of the LCM-MBC implementation, causing the program to halt prematurely, reporting only a portion of the maximal bicliques. The runtime of these two graphs would certainly be much higher if the limit were removed. The results of iMBEA and LCM-MBC on random bipartite graphs are shown in Figure 20. Both methods scale to graphs with thousands of vertices in each partite set. The iMBEA algorithm, however, consistently and convincingly outperforms LCM-MBC.

These figures highlight iMBEA's advantages in scalability. Methods tend not to look very different when graphs are sparse. As data quality improves, however, GeneWeaver and analysis tools of its ilk tend to employ denser graphs in order to capture deeper latent structure. This is where the design enhancements of iMBEA really start to become conspicuous and unmistakable.

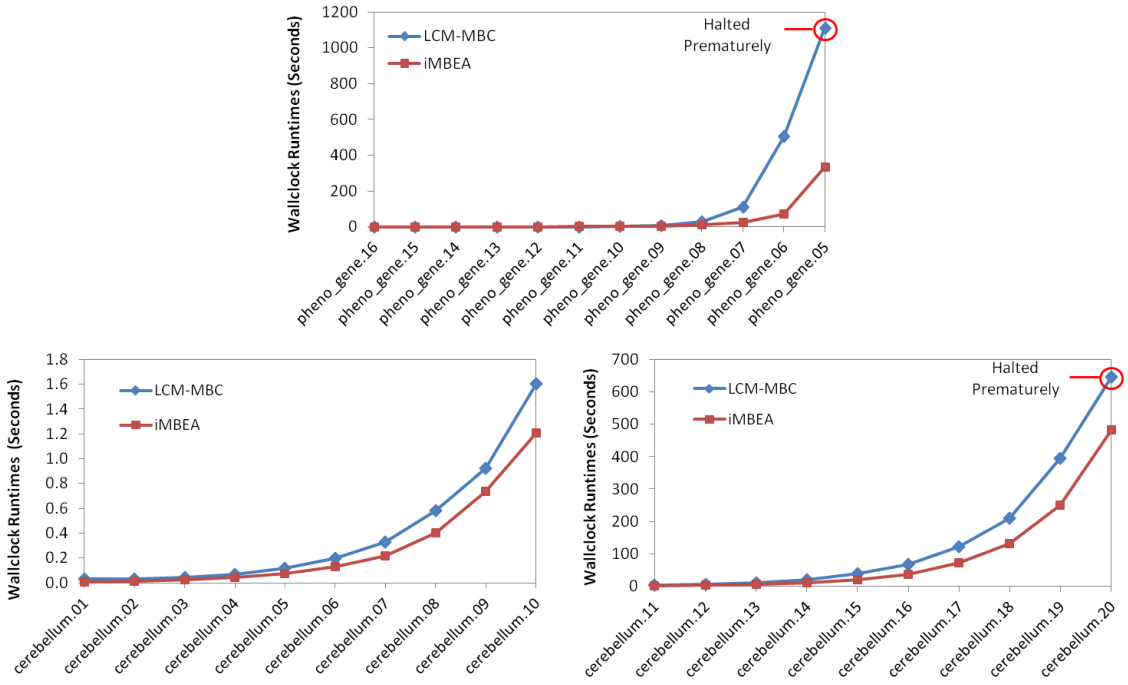

Figure 19. Performance of iMBEA and LCM-MBC on GeneWeaver graphs. The GeneWeaver graphs were constructed from two different phenotype-gene similarity matrices. Each edge is either present or absent based on whether it is at or above (or at or below, when p-values are used) a given threshold. The graphs in the top chart were created from a correlation matrix of 33 phenotypes and 17539 genes. Graphs in the bottom two charts were created from a matrix of correlation p-values for gene expression to phenxotypes in a single mouse population, using 782 phenotypes and 45137 genes. As the threshold moves to the right along the x-axis, the graphs generally grow larger and denser. The pheno-gene graphs range from 6.6% to 34.7% density, while the cerebellum graphs range from about 0.2% to about 2.5% density.

3.2 The Parabiclique Algorithm

Most of this section was previously published in [101]. I designed and implemented the algorithm, using the MBEA algorithm from section 3.1 as a subroutine. I have made numerous minor edits for clarity and to make the prose style and terminology consistent with the rest of this dissertation.

3.2.1 Overview

We present a novel algorithm for extracting dense, disjoint subgraphs from bipartite graphs. Our procedure successively removes such subgraphs, known as parabicliques, by iteratively isolating a maximum biclique and then expanding it in the presence of missing edges. Hence it relies on our previous work on efficiently finding solutions to the *NP*-complete maximum biclique problem. It is also resilient

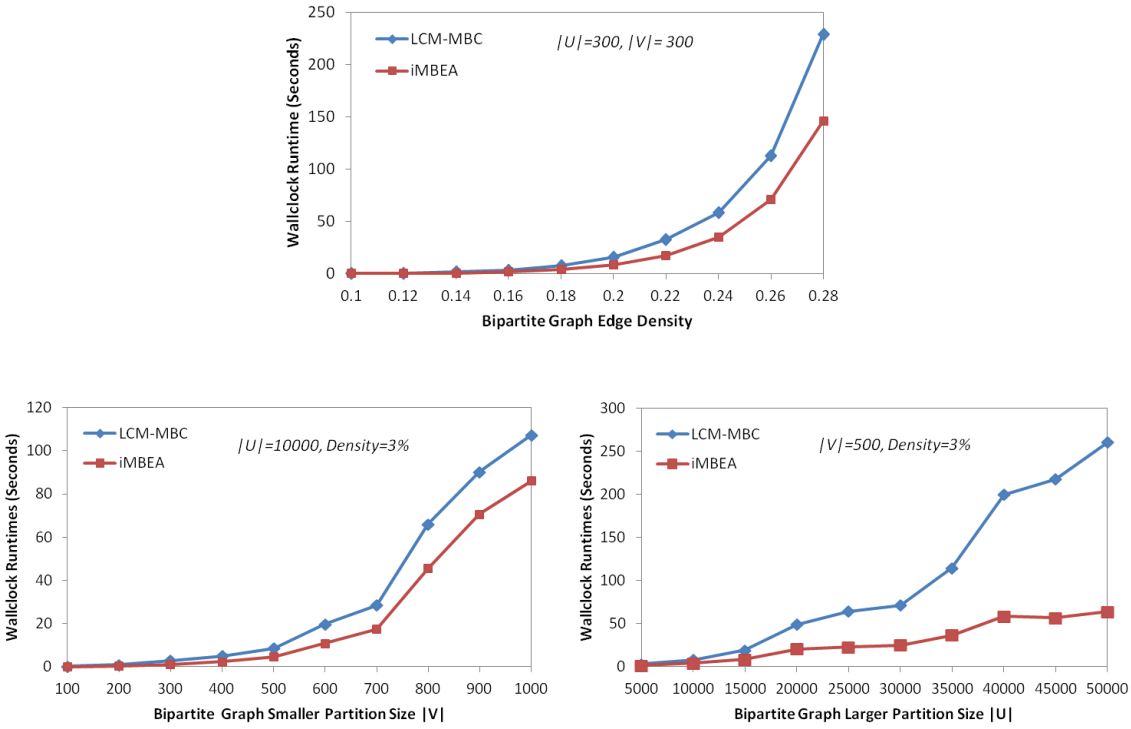

Figure 20. Performance of iMBEA and LCM-MBC on random bipartite graphs. The Erdős-Rényi random bipartite graphs in the top chart have the number of vertices in each partite set fixed at 300, but the density is varied from 0.1 to 0.28, showing how density affects runtime. Similar results on graphs with each partite set fixed at 400 and 500 vertices are omitted for space considerations. The graphs in the bottom two charts were generated using the random graph generator described in the text, with *CV* of 1.0 and density of 0.03. In the bottom left chart, the size of the larger partite set is fixed at 10,000 while the size of the smaller partite set is varied. In the bottom right chart, the converse occurs; the size of the smaller partite set is fixed at 500 while the size of the larger partite set is varied. In all three cases, the performance disparity between iMBEA and LCM-MBC is apparent.

to noise in the form of outliers, poorly correlated raw data and so forth. We have implemented the algorithm and tested it on heterogeneous biological graphs that represent, among other things, associations between genes and diseases, phenotypes, and even microbes. This approach to biological data analysis can be employed as a tool for discovering, confirming and hypothesizing the many roles of genes, gene products and a wide variety of other biological network agents.

3.2.2 Background

Bipartite graphs provide a natural way to model associations between pairs of heterogeneous object classes. Maximally connected subgraphs in bipartite graphs, called bicliques, have proved useful in a huge array of application domains, from

computational biology [102, 103] to wireless networks [104]. Hosts of algorithms address the related problem of biclustering, or co-clustering, in which the rows and columns of a matrix are clustered simultaneously [105, 106]. A two-dimensional matrix can be interpreted as a weighted bipartite graph, and so finding bicliques in undirected graphs can be viewed as a special case of biclustering in which matrix entries are binary.

Here we extend the paraclique notion to the problem of effective bipartite graph clustering. Informally, a parabiclique is a maximum biclique augmented with additional vertices that preserve high but not perfect density. As with paraclique, the main motivations are to decompose highly overlapping edge sets and provide effective data clustering in the presence of noise.

3.2.3 Edge Maximum and Vertex Maximum Bicliques

A *maximal* biclique B is one to which no vertex can be added to form a larger biclique. That is, B is not properly contained in any other biclique. A *maximum* biclique is the largest biclique in a graph. For both maximal and maximum bicliques, we must distinguish between two variants: vertex-maximal (or maximum) and edge-maximal (or maximum). In the former, the size of a biclique is its number of vertices; in the latter, the size is the number of edges. Figure 21 illustrates the difference. More telling from an algorithmic standpoint, the vertex-maximum biclique in a graph can be found in polynomial time, but the problem of finding the edge-maximum biclique is *NP*-complete [87].

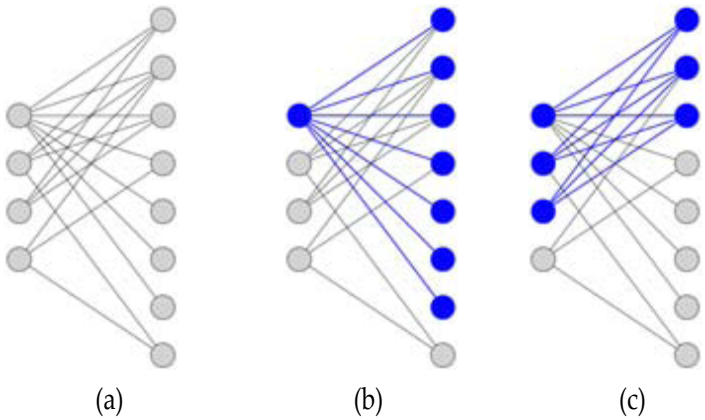

Figure 21. The difference between vertex maximum and edge maximum. The bipartite graph in (a) is shown in (b) with its vertex-maximum biclique highlighted in blue and in (c) with its edge-maximum biclique highlighted in blue.

In practice, edge-maximal biclique is generally the more interesting of the two problem variants. It is probably not surprising that it is the *NP*-complete problem that is the more useful. Edge-maximal bicliques tend to contain multiple vertices from each class, and thus have a higher ratio of edges to vertices and more relationships per vertex. In the domain of biological data analysis, this translates to more relevant molecular response networks and other sorts of putative functional modules. Fortunately, an asymptotically efficient algorithm for enumerating all edge-maximal bicliques, and one that we will use in our implementation of parabiclique, can be found in section 3.1, as published in [72].

3.2.4 Parabiclique Algorithm

We begin by finding a maximum biclique, B. Every vertex not contained in B is then evaluated. If a vertex has sufficient connectivity to B, it is added to B. Otherwise it is discarded. Connectivity to B is generally determined using two parameters, g and h, one for each vertex class. As with the paraclique algorithm, these are either *glom terms* or *proportional glom factors*. In the former case they denote the number of edges allowed to be missing; in the latter case they denote the proportion of edges that must be present. Algorithm 3 depicts g and h as proportional glom factors.

In order to handle cases for which the maximum biclique contains only a few representatives from one class, we sometimes include two additional parameters, w and x, which specify the minimum number of vertices a maximum biclique must contain from each class. (If the biclique contains fewer vertices, then vertices from the other class are not considered for inclusion.) Adjusting parameters g, h, w, and x allows the algorithm to be fine-tuned to bipartite graphs from different application domains. As just one example, testing gene-geneset graphs from GeneWeaver [86] revealed that a proportional glom factor of 0.25 was not unreasonably low, because two genesets having that proportion of common genes are significant. Figure 22 displays an example of a parabiclique.

For efficiency and scalability, we employ the powerful biclique enumeration algorithm, MBEA, as described in [72], to find a maximum biclique. Once a starting maximum biclique has been identified, vertex connectivity computations require at most quadratic time.

Preliminary applications have been revealing. In gene-geneset graphs from GeneWeaver, for example, we have been able to identify novel and potentially revealing associations between disparate experiments using this algorithm.

Algorithm 3: Parabiclique.

Input: A bipartite graph G with partite sets U and V, proportional glom factors g and h, and parameters w and x

Output: A parabiclique, P

```
1   P = B = Maximum biclique in G, with partite sets W ⊆ U and
    X ⊆ V
2   if |W| ≥ w
3     for each v ∈ V \ X
4       if v is connected to at least g|W| vertices in W
5         P = P ∪ {v}
6   if |X| ≥ x
7     for each v ∈ U \W
8       if v is connected to at least g|X| vertices in X
9         P = P ∪ {v};
10  return P;
```

Algorithm 3: Extracting a single parabiclique from a bipartite graph G. Disjoint parabicliques are iteratively extracted by setting $G = G \setminus P$ with each successive call to the algorithm. Iteration continues until some stopping condition is reached, typically when a predetermined number of parabicliques is extracted or when $|B|$ falls below some value. Shown is the proportional glom factor variant. To use glom terms instead, replace $g|X|$ and $g|W|$ with $|X| - g$ and $|W| - g$ respectively.

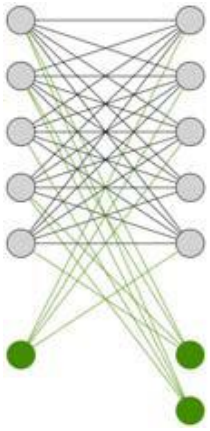

Figure 22. A parabiclique. Vertices of the maximum biclique are in grey. Green vertices are missing just one edge to vertices in the opposing class, thus are included in the parabiclique when $g \geq 1$.

Chapter 4 Algorithms for Multipartite Graphs

A preliminary version of this chapter was presented at the 2nd ACM International Workshop on Big Data in Life Sciences (*BigLS* 2015) [107]. Here I have extended both the theoretical results and applications discussion, including full proofs of the theoretical results, where proof sketches were presented in the conference paper due to space limitations.

Functional genomics, the effort to understand the role of genomic elements in biological processes, has led to an avalanche of diverse experimental and semantic information defining associations between genes and various biological concepts across species and experimental paradigms. Integrating this rapidly expanding wealth of heterogeneous data, and finding consensus among so many diverse sources for specific research questions, require highly sophisticated big data structures and algorithms for harmonization and scalable analysis. In this context, multipartite graphs can often serve as useful structures for representing questions about the role of genes in multiple, frequently-occurring disease processes. The main focus of this chapter is on developing and analyzing an efficient algorithm for dense subgraph enumeration in such graphs. Theoretical results include showing that the tight upper bound of $3^{n/3}$ on the number of maximal cliques in a graph also applies to the number of maximal k-cliques in k-partite graphs for all $k \geq 3$. Our enumeration algorithm has time complexity $O(3^{n/3})$ and therefore realizes the best possible asymptotic behavior. We also give a new proof that finding a vertex-maximum 3-clique in a 3-partite graph is *NP*-hard, and extend it to show *NP*-hardness for finding a vertex-maximum k-clique in a k-partite graph. We also describe and test two problem-reduction heuristics for the algorithm. Empirical testing on both real and synthetic data demonstrates the algorithm's performance. We also describe concrete applications to biological data and scalability issues in the context of big data analysis.

The rest of this chapter is organized as follows. Section 4.1 provides background on clique enumeration and the challenge of data integration in functional genomics. In section 4.2, we discuss vertex and edge maximum k-partite cliques and prove that finding a vertex-maximum k-partite clique in a k-partite graph is *NP*-hard for all $k \geq 3$. In section 4.3, we discuss how biological data is integrated and mapped onto multipartite graphs. In section 4.4, we provide a specific example of modeling query results from a GeneWeaver data integration tool as a multipartite graph, applying our maximal k-partite clique enumeration algorithm, and interpreting the results. In section 4.5, we give an upper bound on the number of maximal k-cliques in a k-partite graph for all $k \geq 3$, and show that the

bound is asymptotically tight. In section 4.6, we present an algorithm, BK-K, to enumerate all maximal *k*-cliques in a *k*-partite graph. Section 4.7 addresses a specific type of *k*-partite graph, a set intersection graph, and shows that maximal *k*-partite clique enumeration in such a graph can be accomplished by enumerating all maximal bicliques in an easily constructed corresponding graph. Section 4.8 briefly discussed alternate problem formulations that, with minor modifications to the algorithm, can be solved by BK-K. In section 4.9, we demonstrate the scaling properties of our algorithm via empirical runtime results on selected random and real-world graphs. Section 4.10 describes two preprocessing heuristics and gives empirical results on their performance. And finally, section 4.11 provides a chapter summary.

4.1 Background

Investigation into the biomolecular basis of normal biological processes in health, disease, development, environmental exposure and speciation has accelerated rapidly with the advent of functional genomics. It is now readily feasible to assess the role of genetic variants, genomic features and gene products across biological systems and states, resulting in many large sets of data consisting of quantitative or discrete associations of biomolecular endpoints with various functional, biobehavioral or disease-related endpoints. The increasing efficiency of data collection methods across the biological sciences has led to an explosive growth in the size and heterogeneity of the corresponding data sets made available for analysis. Moreover, advances in data exposure and discovery have vastly increased the scale and diversity of potential inputs for analysis.

It is widely understood that diseases often share certain features with one another and with normal biological processes as well. In the human immune system, the allergic response and auto-immune diseases are good examples of this. It is also well known that co-occurring disorders may be attributable to common underlying biology. This is particularly the case for behavioral disorders, in which the overlap among conditions makes differential diagnosis and precise therapeutics highly challenging. Furthermore, it is also a common practice to study the biological basis of disease related endpoints in diverse species, where detailed biological investigation in controlled environments is more readily feasible. Finding consensus among diverse studies of similar diseases within and across species, and understanding the conditions under which specific results vary across studies, requires the large-scale integration, comparison and contrast of greatly diverse and very often massive data sets. For example, the last decade has seen the development of platforms capable of simultaneous quantification of DNA methylation at over

450,000 sites, arrays targeting sequences at the exon level with more than 1.2 million probes, and the emergence of RNA-sequencing technologies producing hundreds of millions of reads per study. Genetic mapping analyses implicate variants in disease across the entire genome, and high-content screens examine mutation and drug effects over large numbers of compounds and molecular endpoints.

A major challenge is that diversity of data collection methods applied to particular disease questions vary over time and across investigators, and make for very sparse data around specific research areas, genomic features and experimental paradigms. This problem can be addressed through data harmonization techniques. For example, the alignment of gene products across species through homology, allows the construction of a large matrix of gene × experiment associations [108]. Another complication is that many of the problems we wish to solve are difficult even on data sets of moderate size.

The specter of big data naturally makes the challenge all the more formidable. In [109], we described our work on highly scalable maximum clique solvers for genuinely large graphs, that is, graphs so large they will not fit within core memory. Using our algorithms on Darter, a Cray XC30 at Oak Ridge National Laboratory, we were able to solve the maximum clique problem on graphs built from the California road network with as many as 1.9 million nodes and 2.7 million edges in 153 seconds.

The multipartite enumeration algorithm presented herein is extensible to a wide range of problems in big data analytics. Bipartite graphs are widely used to model data from two types of heterogeneous entities. Adding additional data types naturally extends the model. As a general framework, one only needs the notion of association, similarity or its inverse, difference or distance, between each pair of nodes belonging to different data types. When nodes denote sets containing elements from the same superset, Jaccard similarity can be employed to estimate similarity between members of different partite sets. Diverse similarity measures may be used within the same graph among different partite sets. For instance, two partite sets consisting of nodes representing genesets from two different disease terms could use Jaccard as the similarity metric; while a third partite set representing ontological categories could use enrichment scores.

There is vast untapped potential in relational databases containing extensive collections of biological and biomedical data, and few algorithmic approaches to extract and analyze global similarity of related data resources. Relational databases can be mapped onto multipartite graphs as well, tables being the partite sets, rows the nodes, and edges a relation between rows in different tables, such as foreign keys.

In recent work, we demonstrated the use of clique enumeration for identifying dense networks of co-expressed genes and differentially co-expressed genes [36] and the use of maximal biclique enumeration to enable a data driven classification of disease related experiments based only on the intersection of experimental results provided in the form of genesets [72]. This algorithm has been implemented in the web service, GeneWeaver [86], which enables users to apply bipartite graph analysis to collections of genesets selected from a large database. Here we explore the use of semantic information about the experiments, including the disease concepts or other descriptive text found in geneset metacontent, to enable the comparison of experimental results grouped into categories representing distinct concepts or diseases, and represented as a multipartite graph. To find genes related to the shared biological mechanisms underlying multiple concepts or diseases, we enumerate maximal k-partite cliques in this graph.

While determining whether a graph is bipartite can be done in polynomial time, in general determining whether a graph is k-partite is NP-complete for all $k \geq 3$. The complexity is inherited from the related problem of graph coloring, in that a graph is k-partite if and only if it is k-colorable.

A result of Moon and Moser [39], proven independently by Miller and Muller [110], is that the maximum number of maximal cliques in a graph with n vertices is $3^{n/3}$. A simpler proof of this bound for maximal independent sets can be found in [111].

The classic algorithm for enumerating all maximal cliques in a graph is due to Bron and Kerbosch [37]. The time complexity of this algorithm was shown in [40] to be $O(3^{n/3})$, and is thus asymptotically optimal. Much recent work has been devoted to the enumeration of maximal bicliques [72], partly because the problem is equivalent to the data mining problem of enumerating closed frequent itemsets in transactional data [91]. In the present work we give theoretic results on the problem of enumerating k-partite cliques in k-partite graphs, as well as an asymptotically optimal algorithm.

When, even after filtering and preprocessing, the remaining multipartite graph is too large to fit in core memory, the algorithm presented here would then be mapped onto a big data solution, using out-of-core techniques such as those we devised in [64]. Such direct approaches, while more time-consuming from an implementation standpoint, have numerous advantages in flexibility and performance over simple general-purpose implementations of MapReduce [112], such as Hadoop [113].

4.2 Vertex and Edge Maximum

The size of a clique can be measured by either the number of vertices or the number of edges. The number of edges in an n-clique is $\frac{n(n-1)}{2}$, which is a monotonically increasing function of n when $n \geq 1$, so both vertex and edge measures give equivalent results when determining whether one clique is smaller, the same size, or larger than another clique. But a k-partite clique $K_{n_1,\ldots,n_k}$ has $\sum_{i=1}^{k} n_i$ vertices and $\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} n_i\, n_j$ edges, so the size of one k-partite clique compared to another can differ depending on whether vertices or edges are used as a measure. Consider, for instance, $K_{5,1}$ and $K_{3,2}$. The former has more vertices, but the latter has more edges. Figure 23 illustrates the difference in a 3-partite graph. We must therefore distinguish between edge and vertex measures when considering maximum and maximal k-partite cliques.

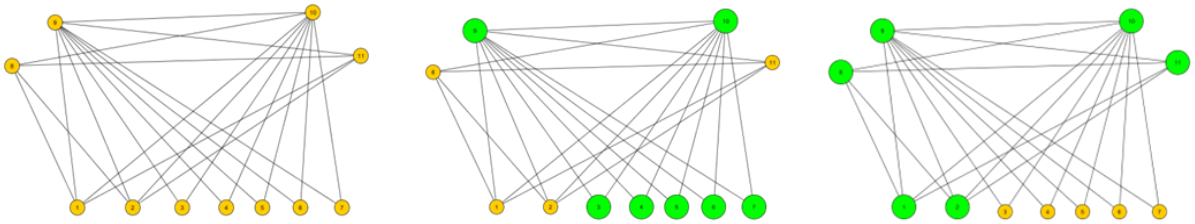

Figure 23. Different results from different measures of k-partite clique size. A 3-partite graph is shown on the left. Its vertex-maximum 3-partite clique is highlighted in green in the middle graph, and its edge-maximum 3-partite clique is shown in green in the graph on the right.

The choice of metric has major algorithmic consequences. A prominent example is that, in a bipartite graph, we can find a vertex-maximum biclique in polynomial time [26], while finding an edge-maximum biclique is *NP*-hard [87]. It is worth noting that showing *NP*-hardness for finding the edge-maximum biclique also resolved, perhaps unknowingly, a conjecture in quadratic programming, stated in [114], that the problem of minimizing a product of linear functions is *NP*-hard. Showing *NP*-hardness for edge-maximum biclique would actually prove a stronger version of the conjecture, namely that it is true even when values are binary.

We now consider the complexity of finding a vertex-maximum k-partite clique in a k-partite graph. As mentioned above, when k = 2, there is a known polynomial time algorithm. But when k = 3, the problem becomes *NP*-hard. Our proof uses a reduction from 1-in-3 SAT to 3-partite independent set. It is a simplified version of a reduction given in [115], where it was used to show approximation results rather than *NP*-hardness. We provide a new proof of correctness for our simplified reduction. The original proof relies on approximation ratios, and appears

to contain a flaw. We then extend the proof to show *NP*-hardness on all *k*-partite graphs for $k \geq 3$.

Theorem 1: Finding a vertex-maximum *k*-partite clique in a *k*-partite graph is *NP*-hard for all $k \geq 3$.

Proof: We reduce a version of 1-in-3 SAT to 3-partite independent set, and thus to the complementary 3-partite vertex-maximum clique problem. 1-in-3 SAT is a variant of 3-SAT that asks, given a Boolean formula in 3CNF, if it has a satisfying assignment such that each clause has exactly one true literal. 1-in-3 SAT was shown to be *NP*-hard in [26], even when restricted to a version where no clause contains a negated literal. Note that we are reducing a decision problem to a decision problem.

Let ϕ be a 1-in-3 SAT instance with no negated literals, where $\phi = C_1 \wedge \ldots \wedge C_m$ and each clause $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$. We reduce ϕ to an instance G of 3-partite independent set, where $G = (X \cup Y \cup Z, E)$ and $|X| = |Y| = |Z| = 3m$, such that ϕ is satisfiable if and only if G has an independent set of cardinality $4m$. The reduction proceeds as follows.

For each clause $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$ in ϕ, we add nine vertices to G: $u_{i,1}$, $u_{i,2}$, $u_{i,3}$, $v_{i,1}$, $v_{i,2}$, $v_{i,3}$, $c_{i,1}$, $c_{i,2}$, and $c_{i,3}$, and we add the following 12 edges to this nine-vertex subgraph:

$$(u_{i,1}, v_{i,1}),\ (u_{i,2}, v_{i,2}),\ (u_{i,3}, v_{i,3}),$$
$$(c_{i,1}, u_{i,2}),\ (c_{i,1}, u_{i,3}),\ (c_{i,1}, v_{i,1}),$$
$$(c_{i,2}, u_{i,1}),\ (c_{i,2}, u_{i,3}),\ (c_{i,2}, v_{i,2}),$$
$$(c_{i,3}, u_{i,1}),\ (c_{i,3}, u_{i,2}),\ (c_{i,3}, v_{i,3}).$$

Figure 24 shows the resulting subgraph.

Next, for each pair of identical literals in different clauses, l_{is} and l_{jt}, $i \neq j$, we add the following six edges to G:

$$(u_{i,s}, c_{j,h}), 1 \leq h \leq 3, h \neq t \text{ (2 edges)}$$
$$(c_{i,h}, u_{j,t}), 1 \leq h \leq 3, h \neq s \text{ (2 edges)}$$
$$(v_{i,s}, c_{j,t}),\ (c_{i,s}, v_{j,t}) \text{ (2 edges)}$$

Figure 25 shows an example.

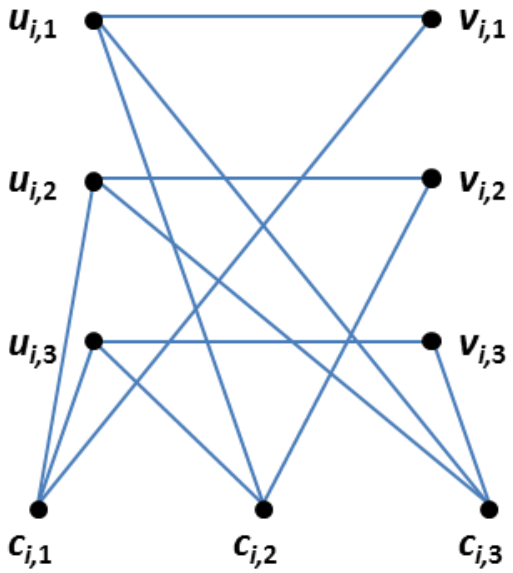

Figure 24. A subgraph representing one clause of a 1-in-3 SAT instance. Such a subgraph, C_i, is produced for each clause of the 1-in-3 SAT instance in the reduction to an instance of 3-partite independent set. Vertices $u_{i,1}$, $v_{i,1}$ and $c_{i,1}$ represent literal $l_{i,1}$; vertices $u_{i,2}$, $v_{i,2}$ and $c_{i,2}$ represent literal $l_{i,2}$; and vertices $u_{i,3}$, $v_{i,3}$ and $c_{i,3}$ represent literal $l_{i,3}$.

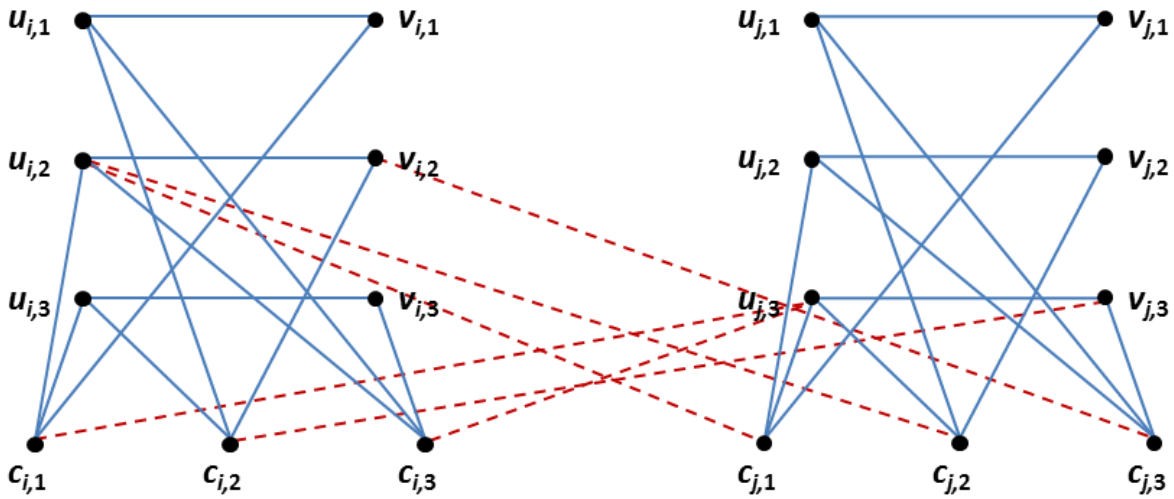

Figure 25. Edges added between two subgraphs, each representing a clause. Six edges are added between subgraphs for each pair of identical literals in different clauses. In the above example, $l_{i,2}$ and $l_{j,3}$ are identical literals appearing in clauses C_i and C_j, prompting the addition of the six red dashed edges. If a pair of clauses has two literals in common, then 12 edges will be added between the two respective subgraphs.

The construction of G is now complete. It has $9m$ vertices and $12m + 6a$ edges, where a is the number of identical pairs of literals in different clauses. Since all pairs of clauses must be compared for identical literals, the reduction has time complexity $O(m^2)$.

Observe that all the u vertices are independent, all the v vertices are independent, and all the c vertices are independent. Now define $X = \{u_{i,j} \in V(G)\}$, $Y = \{v_{i,j} \in V(G)\}$, $Z = \{c_{i,j} \in V(G)\}$. Then G is a 3-partite graph with partite sets X, Y and Z, each with $3m$ vertices.

Claim: ϕ is satisfiable if and only if G has an independent set of cardinality $4m$.

Proof: We first prove the reverse (only if) implication. Suppose ϕ is satisfiable. Then given a satisfying assignment A of ϕ, we can construct an independent set I of size $4m$ in G by doing one of the following for each clause C_i:

1. If $l_{i,1}$ is the one literal set to true in A, add vertices $c_{i,1}$, $u_{i,1}$, $v_{i,2}$, and $v_{i,3}$ to I.
2. If $l_{i,2}$ is the one literal set to true in A, add vertices $c_{i,2}$, $u_{i,2}$, $v_{i,1}$, and $v_{i,3}$ to I.
3. If $l_{i,3}$ is the one literal set to true in A, add vertices $c_{i,3}$, $u_{i,3}$, $v_{i,1}$, and $v_{i,2}$ to I.

Each group of four added vertices is an independent set. And if two clauses C_i and $\underline{C_j}$ do not share at least one identical literal, then of course no pair of vertices from different groups will be adjacent. Examining all the possible ways that C_i and $\underline{C_j}$ can share at least one identical literal shows that no two vertices can be adjacent. Therefore I is an independent set of cardinality $4m$.

We now prove the forward (if) implication. Suppose G has an independent set I of cardinality $4m$. We show how to construct a satisfying assignment of ϕ from I.

First, observe that each nine-vertex induced subgraph of G corresponding to a clause C_i has three maximum independent sets of cardinality 4: $\{c_{i,1}, u_{i,1}, v_{i,2}, v_{i,3}\}$, $\{c_{i,2}, u_{i,2}, v_{i,1}, v_{i,3}\}$ and $\{c_{i,3}, u_{i,3}, v_{i,1}, v_{i,2}\}$, so that each subgraph can contribute at most four vertices to I. Therefore, to achieve the assumed $4m$ cardinality, I must include exactly four vertices from each nine-vertex subgraph, exactly one of which is a c vertex.

To construct an assignment A for ϕ, we set, for each clause C_i, the literal $l_{i,s}$ to true when vertex $c_{i,s}$ is included in I. We set the other two literals in C_i to false.
We claim that A is a valid assignment for ϕ. Observe that exactly one literal has been set to true in each clause. Therefore, as long as there are no conflicts (a literal set to true in one clause but false in another) the assignment is valid.

Assume A has a conflict between identical literals $l_{i,s}$ and $l_{j,t}$ where $i \neq j$. Without loss of generality, say $l_{i,s} = 1$ and $l_{j,t} = 0$. This means that $c_{i,s} \in I$ and $c_{j,t} \notin I$. Thus, one of the two vertices in $\{c_{j,1}, c_{j,2}, c_{j,3}\} \backslash \{c_{j,t}\}$ must be in I. But $c_{i,s} \in I$ implies $u_{i,s} \in I$, and $u_{i,s}$ is adjacent to both vertices in $\{c_{j,1}, c_{j,2}, c_{j,3}\} \backslash \{c_{j,t}\}$ because $l_{i,s}$ and $l_{j,t}$ are identical literals. We have contradicted the assumption that I contains two adjacent vertices. Therefore A does not have any conflicts, and is a valid assignment for ϕ.

Having reduced 1-in-3 SAT to 3-partite independent set, and thus to 3-partite vertex-maximum clique, we have shown that finding a vertex-maximum 3-partite clique in a 3-partite graph is *NP*-hard. The results readily generalize to the problem of finding a vertex-maximum *k*-partite clique in a *k*-partite graph as follows.

Given a 3-partite graph *G*, we can construct an arbitrary *k*-partite graph, $k > 3$, by adding $k - 3$ vertices, each connected to all other vertices in the graph. Each new vertex is the sole member of a new partite set. A *k*-partite clique in *G′* is vertex-maximum if and only if it consists of a vertex-maximum 3-partite clique in *G* and all $k - 3$ added vertices.

Thus we have shown that finding a vertex-maximum *k*-partite clique in a *k*-partite graph is *NP*-hard for all $k \geq 3$. □

4.3 Exploratory Data Integration Using Multipartite Graphs

An unweighted *k*-partite graph can be constructed from any *k* sets of objects where there is a similarity (or distance) measure between any two objects in different sets. The application of a threshold to the similarity value between two objects results in either an edge or a non-edge. As in the set-set graph, the objects themselves can be sets, in which case we can apply a similarity metric appropriate to set comparisons, such as Jaccard similarity.

For example, a set of experiments represented in a gene x experiment association matrix can be divided into submatrices based on semantic content, e.g., relevance to different co-occurring diseases. The resulting adjacency matrix can thus be represented as a *k*-partite graph consisting of one partite set containing genes and one additional partite set for each semantic term describing a set of genesets. See Figure 26.

From an application perspective, the *k*-partite graph representation of biological associations has a broad potential impact. Virtually any triple store can be represented as a 3-partite graph, enabling the comparison of triples that feature a common entity. Thus, many bioinformatics data resources, including data from model organism databases, MEDLINE abstracts, Electronic Medical Records (EMR),

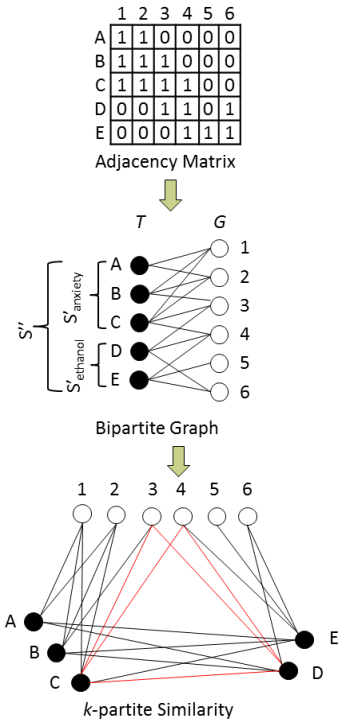

Figure 26. Construction of a *k*-partite graph by partitioning one partite set of a bipartite graph (*k*=3). Maximal *k*-partite cliques like the triclique denoted by red edges contain nodes from every partite set.

chemoinformatics resources, and a host of drug-gene-disease databases including the Comparative Toxicogenomics Database (CTD) [116] can be analyzed simultaneously using this approach. To facilitate this, we have mapped many of these data resources onto one another in the GeneWeaver system. The federation of closely related biological databases may represent a relatively small data size, but the strategy can achieve incredible integrative effects at the level of precision medicine [117, 118].

EMR mining is another promising application area for *k*-partite graph applications in big data analytics. For example, EMR record integration through *k*-partite analysis can be used to aggregate individual data from clinical sequencing and laboratory procedures into much larger resource data sets, thereby allowing physicians to make use of evidence-based medicine within much smaller subpopulations, e.g. how does a single patient with a history of alcohol abuse present within the context of a larger population of patients with similar behavioral backgrounds, clinical findings, and disease markers. More importantly, however, this area highlights remaining work in the practical implementation of multipartite graph analytics within the constraints provided by ever increasing demands on

scalable computing. EMR big data advocates argue the need for boundless aggregation of all applicable data for use with appropriate analytic algorithms, requiring that exhaustive measures of *k*-partite sets must be executed within a landscape of constantly changing data density and size. One means that big data provides to address this issue is to encourage partial computation of smaller, fault-tolerant data sets. Examining how our approach can be applied within scalable cloud computing environments using the aggregation of smaller, possibly pre-computed, *k*-partite sets into larger maximal *k*-cliques, for example, will allow us to explore the dynamic management of time sensitive queries.

4.4 Multipartite Data Integration Example

As a sample application we constructed a 3-partite graph. Vertices in one partite set represent genesets containing the terms alcohol or ethanol in their descriptive metacontent. (Alcohol is frequently used by researchers in alcoholism, where it is intended to refer to ethanol or ethyl alcohol.) Vertices in the second partite set represent genesets associated with stress or anxiety in their descriptive metacontent. Vertices in the third partite set represent the genes that are the elements in these sets. Jaccard similarity was calculated between each pair of vertices in different partite sets representing genesets, then thresholded to place either an edge or non-edge. Edges between vertices in the partite set representing genes and vertices in the other partite sets represent a gene's membership in a geneset.

Using the *k*-partite clique enumeration algorithm, genes from genesets related to alcohol or ethanol and anxiety or stress, were analyzed. The search for the terms alcohol or ethanol and anxiety or stress yielded a graph with 836 x 264 x 32093 vertices and 394,789 edges. See Figure 27. The graph had 79,998 maximal tricliques, which were enumerated in a runtime of 81.58 seconds. The maximal tricliques were then filtered, removing those with only one vertex in a partite set, a process analogous to requiring a minimum support for itemsets in data mining. The result was 39,586 filtered maximal tricliques. One triclique contained two genesets related to alcohol/ethanol and seven sets related to anxiety/stress. The alcohol or ethanol related sets contain genes annotated to the medical subject heading term alcoholism (GS128735) and genes encoding protein biomarkers of alcohol abuse (GS216653) [119]. The stress related sets contained genes annotated to the Gene Ontology terms "response to stress" (GS193563, GS210507), "cellular response to stress" (GS180482, GS197275), "regulation of response to stress" (GS190410, GS207306) in mouse and human, as well as genes experimentally shown to be differentially expressed in the nucleus accumbens brain region between high responding and low responding lines of selectively bred rats (GS135132) [120]. The 3-clique contained four gene vertices

labeled *Tnf, Il1b, Il1a* and *Il6.* Bioinformatics software tools were used to assess the putative function of this group of genes. KEGG-Pathway enrichment analysis shows all four genes within a cytokine-cytokine receptor interaction pathway (Benjamini FDR adjusted $p=6.8 \times 10^{-4}$). BIOCARTA indicated three are found in signaling through the IL1R pathway (FDR adjusted p=0.01). These results suggest that the relationship of alcoholism and stress response is associated with cytokine response, a neuroimmune mechanism. Many studies have shown a role for this neuroimmune system in behavior, specifically alcohol and drug addiction [121, 122], as well as neuropsychiatric disorders such as depression and anxiety [123, 124].

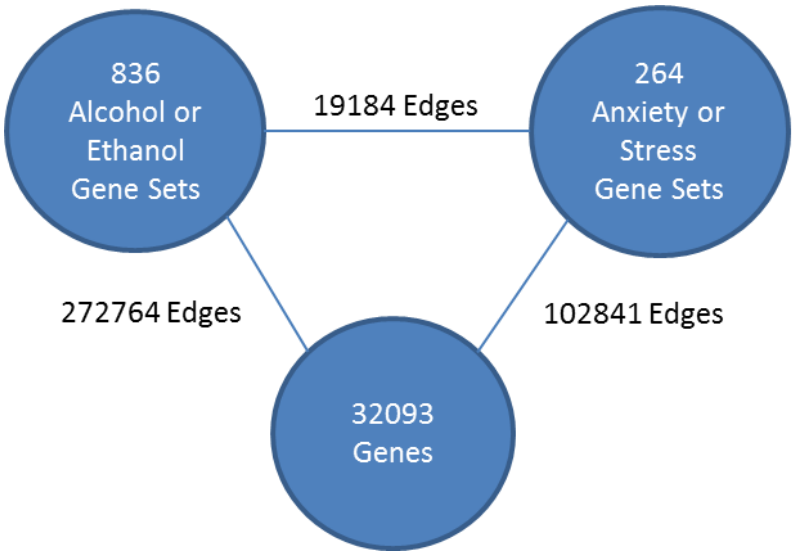

Figure 27. A tripartite graph constructed from GeneWeaver data. Two partite sets contain genesets returned by queries. The third partite set contains all genes in the returned genesets. This graph contained 79998 maximal tricliques.

4.5 An Upper Bound on the Number of Maximal *k*-partite Cliques

As mentioned, the number of maximal cliques in a graph has a tight upper bound of $3^{n/3}$. The same tight bound of $3^{n/3}$ was shown for the number of maximal bicliques in a general graph [125], which differs from the upper bound of $2^{n/2}$ on the number of maximal bicliques in a bipartite graph. Here we show that the number of maximal *k*-partite cliques in a *k*-partite graph has an asymptotically tight bound of $3^{n/3}$ for all $k \geq 3$. Our proof uses techniques similar to those used by Moon and Moser.

First, we make the following observation, which is also key to our modification of the BK algorithm in section 4.6.

Observation 1. If G is a k-partite graph, and $G' = (V', E')$ is constructed by adding all intrapartite edges to G, then $C \subseteq V$ is a maximal clique in G' if and only if $C' \subseteq V'$ is a maximal k-partite clique in G, where C'contains the same vertices in G' as C in G.

That is, any maximal k-partite clique in G is a maximal clique in G'. We use this observation in the following theorem.

Theorem 2. For all $k \geq 3$, the maximum number of maximal k-partite cliques in a k-partite graph of order n is ~ $3^{n/3}$.

Proof: By Observation 1, every maximal k-partite clique in G is a maximal clique in a graph G' constructed by adding all possible intrapartite edges. Therefore the upper bound of $3^{n/3}$ for the number of maximal cliques in a graph applies to the number of k-partite cliques in k-partite graphs.

To show that the upper bound is asymptotically tight, we construct balanced 3-partite, 4-partite, and 5-partite graphs, each with $\Omega(3^{n/3})$ maximal k-partite cliques. We then show that the constructions extend to the general k-partite case. For each fixed k, we construct a balanced k-partite graph G of order n (i.e. each partite set has n/k vertices) such that the number of maximal k-partite cliques is at least $g(n)$, where $g(n)$ is a function of k and n based on the specific construction. On the other hand, based on a classic result of Moon and Moser, a graph of order n can have at most $F(n) = 3^{n/3}$ maximal cliques. It turns out that $g(n)$ and $F(n)$ grow asymptotically on the same order as n tends to infinity no matter the value of k. Thus the maximum number of maximal k-partite cliques in a k-partite graph is asymptotically the same order as the maximum number of maximal cliques of a general graph with the same number of vertices. □

Proof: We construct balanced 3-partite, 4-partite, and 5-partite graphs, each with $\Omega(3^{n/3})$ maximal k-partite cliques. We then show that the constructions extend to the general k-partite case.

3-partite. Let $G = (X \cup Y \cup Z, E)$ be a tripartite graph where $|X| = |Y| = |Z| = n$. Suppose $X = \{x_1, x_2, \dots, x_n\}$, $Y = \{y_1, y_2, \dots, y_n\}$ and $Z = \{z_1, z_2, \dots, z_n\}$. Let $E = \{(x_i, y_j) | i \neq j\} \cup \{(x_i, z_k) | i \neq k\} \cup \{(y_j, z_k) | j \neq k\}$. That is, G is the result of removing (the edges of) n disjoint triangles $\{(x_i, y_i, z_i) | i = 1, \dots, n\}$ from a balanced complete tripartite graph.

Now consider any partition I, J, K of the sequence $\{1,2, \dots, n\}$. That is, I, J and K are nonempty subsets of $\{1,2, \dots, n\}$, they are pairwise disjoint and their union is $\{1,2, \dots, n\}$. Observer that $A \subseteq X, B \subseteq Y$ and $C \subseteq Z$, with $A = \{x_i | i \in I\}, B = \{y_j | j \in J\}$

and $C = \{z_k | k \in K\}$ constituting a maximal triclique (A, B, C) of G. Put another way, there is a bijection between the set of partitions of $\{1,2,\dots,n\}$ and the set of maximal tricliques of G. Any partition I, J, K can be obtained by choosing from $\{1,2,\dots,n\}$ without replacement a elements to be in I, b elements to be in J, and c elements to be in K. Since we require the partite sets I, J, K to be nonempty, we have $0 < a \leq n-2$ and $1 \leq b < n-a$. For fixed a and b, the number of possible choices for I, J, K is $\binom{n}{a}\binom{n-a}{b}$. Thus, the total number of partitions of $\{1,2,\dots,n\}$ is $f(n) = \sum_{a=1}^{n-2}\sum_{b=1}^{n-a-1}\binom{n}{a}\binom{n-a}{b}$. Simplifying, we have $f(n) = \sum_{a=1}^{n-2}\binom{n}{a}(2^{n-a}-2) = 3^n - 3(2^n - 1)$. We know from Moon and Moser's 1965 paper that a general graph with $3n$ vertices can have a maximum of $F(n) = 3^{n/3}$ maximal cliques. Since $\lim_{n\to\infty}\frac{f(n)}{F(n)} = 1$, we conclude that the number of maximal tricliques of G is ~ $3^{n/3}$.

4-partite. Next we construct a balanced 4-partite graph with $\Omega(3^{n/3})$ maximal 4-partite cliques.

Let $G = (W \cup X \cup Y \cup Z, E)$ be a 4-partite graph where $|W| = |X| = |Y| = |Z| = 3n$. Suppose $W = \{w_1, w_2, \dots, w_{3n}\}$, $X = \{x_1, x_2, \dots, x_{3n}\}$, $Y = \{y_1, y_2, \dots, y_{3n}\}$ and $Z = \{z_1, z_2, \dots, z_{3n}\}$. Define $W_1 = \{w_1, w_2, \dots, w_n\}$, $W_2 = \{w_{n+1}, w_{n+2}, \dots, w_{2n}\}$, $W_3 = \{w_{2n+1}, w_{2n+2}, \dots, w_{3n}\}$ and similar notations for X, Y and Z. Construct G by removing (the edges of) the following $4n$ disjoint triangles from a balanced complete 4-partite graph.

$E_1 = \{(x_i, y_i, w_i) | i = 1, \dots, n\}$, among W_1, X_1, Y_1
$E_2 = \{(z_i, y_{i+n}, w_{i+n}) | i = 1, \dots, n\}$, among W_2, Y_2, Z_2
$E_3 = \{(z_{i+n}, x_{i+n}, w_{i+2n}) | i = 1, \dots, n\}$, among W_3, X_2, Z_2
$E_4 = \{(z_{i+2n}, x_{i+2n}, y_{i+2n}) | i = 1, \dots, n\}$, among X_3, Y_3, Z_3

We obtain a lower bound for $f(n)$, the number of maximal 4-partite cliques in G. There are at least $t(n) = \sum_{a=1}^{n-2}\sum_{b=1}^{n-a-1}\binom{n}{a}\binom{n-a}{b} = 3^n - 3(2^n - 1)$ ways to choose vertices from W_1, X_1, Y_1, from W_2, Y_2, Z_1, from W_3, X_2, Z_2 and from X_3, Y_3, Z_3. Thus, the number of maximal 4-partite-cliques of G is at least $g(n) = (t(n))^4 = (3^n - 3(2^n - 1))^4$. Observe that G has a total of $12n$ vertices and a general graph with $12n$ vertices can have $F(n) = 3^{4n}$ maximal cliques. Since $\lim_{n\to\infty}\frac{g(n)}{F(n)} = 1$, $g(n) \leq f(n) \leq F(n)$, we have that $\lim_{n\to\infty}\frac{f(n)}{F(n)} = 1$. Therefore the number of maximal 4-partite cliques of G is ~ $3^{n/3}$.

5-partite. Next we construct a balanced 5-partite graph with $\Omega(3^{n/3})$ maximal 5-partite cliques.

Let $G = (U \cup V \cup W \cup X \cup Y, E)$ be a 5-partite graph where $|U| = |V| = |W| = |X| = |Y| = 3n$. Suppose $U = \{u_1, u_2, \dots, u_{3n}\}, V = \{v_1, v_2, \dots, v_{3n}\}, W = \{w_1, w_2, \dots, w_{3n}\}$, $X = \{x_1, x_2, \dots, x_{3n}\}$ and $Y = \{y_1, y_2, \dots, y_{3n}\}$. Define $W_1 = \{w_1, w_2, \dots, w_n\}$, $W_2 = \{w_{n+1}, w_{n+2}, \dots, w_{2n}\}$, $W_3 = \{w_{2n+1}, w_{2n+2}, \dots, w_{3n}\}$ and similar notations for X, Y, U and V. Construct G by removing (the edges of) the following $5n$ disjoint triangles from a balanced complete 5-partite graph.

$E_1 = \{(u_i, v_i, w_i) | i = 1, \dots, 2n\}$, among $U_1 \cup U_2$, $V_1 \cup V_2$, $W_1 \cup W_2$
$E_2 = \{(v_{i+2n}, x_{i+n}, y_{i+n}) | i = 1, \dots, n\}$, among V_3, X_2, Y_2
$E_3 = \{(u_{i+2n}, x_{i+2n}, y_{i+2n}) | i = 1, \dots, n\}$, among U_3, X_3, Y_3
$E_4 = \{(w_{i+2n}, x_i, y_i) | i = 1, \dots, n\}$, among W_3, X_1, Y_1

Observe that G has at least $g(n) = (3^{2n} - 3(2^{2n} - 1))(3^n - 3(2^n - 1))^3$ maximal 5-partite cliques. A general graph with $15n$ vertices can have $F(n) = 3^{5n}$ maximal cliques. Since $\lim_{n\to\infty} \frac{g(n)}{F(n)} = 1$, $g(n) \leq F(n)$, the number of maximal 5-partite cliques of G is ~ $3^{n/3}$.

Extension to general k. Now consider a balanced k-partite graph G, with $k \geq 6$. There are three possible cases.

Case 1: $k \equiv 0 \bmod 3$. Suppose each partite set in G has n vertices. We group the partite sets into groups of 3, for a total of $k/3$ groups. Next we remove (the edges of) n disjoint triangles as we did for the tripartite case. In total we remove $kn/3$ disjoint triangles. The resulting graph G' has $f(n) = (3^n - 3(2^n - 1))^{k/3}$ maximal k-partite cliques. A general graph with kn vertices can have $F(n) = 3^{kn/3}$ maximal cliques. Since $\lim_{n\to\infty} \frac{f(n)}{F(n)} = 1$, the number of maximal k-partite cliques of G' is ~ $3^{n/3}$.

Case 2: $k \equiv 1 \bmod 3$. Suppose each partite set in G has $3n$ vertices. We first take four partite sets and remove (the edges of) $4n$ disjoint triangles among them, as we did for 4-partite case. Next we group the remaining partite sets into groups of 3, for a total of $(k-4)/3$ groups. We remove (the edges of) $3n$ disjoint triangles for each group as we did for the tripartite case. In total we remove $4n + \left(\frac{k-4}{3}\right) 3n = kn$ disjoint triangles. The resulting graph G' has at least $g(n) = (3^n - 3(2^n - 1))^4 (3^{3n} - 3(2^{3n} - 1))^{(k-4)/3}$ maximal k-partite cliques. A general graph with $3kn$ vertices can have $F(n) = 3^{kn}$ maximal cliques. Since $\lim_{n\to\infty} \frac{g(n)}{F(n)} = 1$, the number of maximal k-partite cliques of G' is ~ $3^{n/3}$.

Case 3: $k \equiv 2 \bmod 3$. Suppose each partite set has $3n$ vertices. We first take five partite sets and remove (the edges of) $5n$ disjoint triangles among these partite sets as we did for the 5-partite case. Next we group the remaining partite sets into groups of 3, for a total of $(k-5)/3$ groups. We remove (the edges of) $3n$ disjoint

triangles for each group as we did for the tripartite case. The resulting graph G' has at least $g(n) = (3^{2n} - 3(2^{2n} - 1))(3^n - 3(2^n - 1))^3(3^{3n} - 3(2^{3n} - 1))^{(k-5)/3}$ maximal k-partite cliques. A general graph with $3kn$ vertices can have $F(n) = 3^{kn}$ maximal cliques. Since $\lim_{n\to\infty} \frac{g(n)}{F(n)} = 1$, the number of maximal k-partite cliques of G' is ~ $3^{n/3}$.

In summary, for every $k \geq 3$, we have shown that there is a balanced k-partite graph whose number of maximal k-partite cliques is ~ $3^{n/3}$, or asymptotically the same order as the maximum number of maximal cliques of a general graph with the same number of vertices. □

4.6 A *k*-partite Clique Enumeration Algorithm

In 1973 Bron and Kerbosch published two recursive backtracking algorithms for enumerating all maximal cliques in a graph. The first was a basic version; the second one used a pivot vertex. They recognized that the second one was superior, saving many recursive calls in practice, especially on graphs containing large numbers of non-maximal cliques. We shall refer to this second algorithm as the BK algorithm.

The hallmark of the BK algorithm is the use of three dynamically changing vertex sets in a recursive backtracking strategy. The set P contains the current clique, the set R contains vertices that can extend the current clique, and the set X contains vertices that have already been tried. We refer to any minor variant on this basic scheme as a Bron-Kerbosch algorithm.

One major advantage of the BK algorithm is that previously found maximal cliques do not need to be retained in memory, but can be discarded after they are output. Only the graph and vertex sets P, R and X need to be stored. Hence a signature of BK implementations is very low memory overhead above what is necessary to store the graph. Duplicating the same maximal clique is prevented by clever use of the set X. Many subsequent modifications to the BK algorithm have been suggested, most focusing on improving the method for selecting pivot vertices. For instance, the modification in [40] chooses the pivot not just from P, but from $P \cup X$. In [126], the vertex first in a degeneracy ordering is chosen for the pivot, which can result in faster performance on sparse graphs. A BK-based algorithm to enumerate all maximal cliques in order of size was given in [127]. Maximal clique enumeration algorithms using schemes different from that employed by Bron and Kerbosch have been published. For instance, see [35]. But empirical testing has shown that BK-based algorithms outperform alternatives [128].

As far as we are aware, only two algorithms have been published to enumerate k-partite cliques in k-partite graphs. The CLICKS algorithm in [129] frames the problem in terms of categorical data clustering, seeking to enumerate all

subspace clusters, where a subspace cluster is effectively a *k*-partite clique in a *k*-partite graph without the requirement that every partite set have at least one node in the cluster. Such clusters are called full-space clusters when each partite set has at least one node in the cluster. CLICKS is inspired by Bron-Kerbosch and employs a similar recursive backtracking strategy. It holds all discovered maximal *k*-partite cliques in memory and then post-processes each one. (An obvious improvement would quite naturally be to post-process and output each maximal clique when it is discovered, rather than store them all, although this will not materially change the worst-case runtime of the algorithm.) The algorithm in [130] is less efficient. It calls as a subroutine an algorithm to enumerate all maximal bicliques. The subroutine must be called between each pair of partite sets. All subsets of each maximal biclique must then be considered in turn to determine if they can be extended to *k*-partite cliques with $k > 2$. The authors observe that this algorithm scales poorly, but scaling is not the focus of their article.

The modifications we introduce to the BK algorithm so that it enumerates maximal *k*-partite cliques in *k*-partite graphs can be integrated into any maximal clique enumeration algorithm that uses BK-style recursive backtracking. The modifications stem from Observation 1 in the previous section, which states that when adding all possible intrapartite edges to a *k*-partite graph *G* to form a graph *G′*, then any maximal *k*-partite clique in *G* will be a maximal clique in *G′*. Therefore, any algorithm that enumerates all maximal cliques in general graphs can be adapted to enumerate maximal *k*-partite cliques in a *k*-partite graph. Applying Observation 1, we modified the BK algorithm to enumerate all *k*-partite cliques in a *k*-partite graph. The resulting algorithm (Algorithm 4) is called BK-K, for Bron-Kerbosch *k*-partite.

Theorem 3. The time complexity of the BK-K algorithm is $O(3^{n/3})$.

Proof: The complexity of the Bron-Kerbosch algorithm was shown to be $O(3^{n/3})$ in [39]. Our modification adds a one-time preprocessing step to insert all intrapartite edges (line 1) and a check that each maximal clique contains one vertex from each partite set (line 8). The preprocessing step can be performed in $O(n^2)$ time. The check that a maximal clique contains a vertex from each partite set can be performed with *k* array accesses by maintaining a size *k* array storing the number of vertices from each partite set in *R*. If a hash table is used, maintaining the array adds a constant time lookup to each insertion and deletion from *R*. The overall complexity of BK-K is therefore $O\left(k3^{n/3} + n^2\right) = O(3^{n/3})$. □

Algorithm 4: BK-K: k-partite Bron-Kerbosch with pivot.

Input: a k-partite graph, $G = (V, E)$
Output: all maximal k-partite cliques in G

```
1  Add all possible intrapartite edges to G
2  R := ∅, P := V, X := ∅
3  BK-K(R, P, X)
4      if P and X are both empty
5          report R as a maximal clique
6      choose a pivot vertex u in P ∪ X
7      for each vertex v in P\N(u)
8          if R ∪ v ∪ P ∩ N(v) contains at
           least one vertex from each partite
           set
9              BK-K(R ∪ v, P ∩ N(v), X ∩ N(v))
10         P := P\v
11         X := X ∪ V
```

Algorithm 4. A modification (lines 1 and 8) to the Bron-Kerbosch algorithm results in an algorithm to enumerate all maximal k-partite cliques in a k-partite graph.

4.7 Multipartite Set Intersection Graphs

We define a k-*partite set intersection graph* to be a k-partite graph in which vertices in one partite set represent elements and vertices in all other partite sets represent sets. Edges between sets signify that the sets have at least one element in common. Edges between an element and a set signify that the element is a member of the set. Figure 28 shows an example of a 3-partite set intersection graph.

Our definition of a set intersection graph does not require that elements of all sets be represented by vertices in the graph, although such may be the case. Such graphs can be applied to problems in bioinformatics and computational biology when searching for relationships between sets of entities. For example, in the GeneWeaver system, groups of genesets may be selected from a database and represented as a k-partite graph. One may select a group of genesets with some commonality, such as relation to a particular trait or ontology term, and another group of genesets with a different commonality, such as association with a particular disease. One would seek to find subsets of genes in common between the trait or ontology term and the disease. Such an application could be extended to other problems in ontology comparison and cross-mapping.

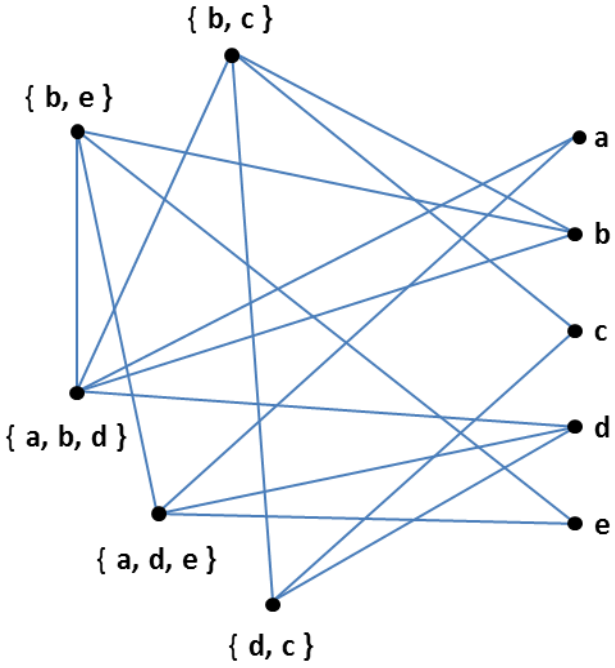

Figure 28. A 3-partite set intersection graph. Vertices $\{b, e\}$ and $\{b, c\}$ comprise one partite set; vertices $\{a, b, d\}$, $\{a, d, e\}$ and $\{d, c\}$ comprise a second partite set; and vertices a, b, c, d and e comprise the third partite set. One partite set always consists of set elements. Interpartite edges indicate either that sets have at least one element in common or that an element is a member of a set. Such graphs have no intrapartite edges.

We have shown that enumerating maximal bicliques in a bipartite graph is not equivalent to enumerating maximal *k*-cliques in *k*-partite graphs. As it turns out, however, when graphs are restricted to *k*-partite set intersection graphs, enumerating maximal *k*-partite cliques can be accomplished by enumerating maximal bicliques in an easily constructed corresponding bipartite graph. Given a *k*-partite set intersection graph *G*, the corresponding bipartite graph G_b is *G* with all edges between sets removed, so that only edges between sets and elements remain. Vertices in G_b that represent elements form one partite set; vertices that represent sets, which in *G* were partitioned into two or more partite sets, form the second partite set in G_b.

Theorem 4. A *k*-partite clique in a *k*-partite set intersection graph *G* is maximal if and only if its corresponding vertices form a maximal biclique in the bipartite graph G_b.

Proof. Let *G* be a *k*-partite set intersection graph and G_b the corresponding bipartite graph. Consider a maximal *k*-partite clique *K* in *G*. Every element vertex in *K* has an

edge to every set vertex in K. Since only edges between set vertices are removed when constructing G_b, this is still true when considering K in G_b. Therefore K is a biclique in G_b. And K is maximal; otherwise K would not be maximal in G. Now consider a maximal biclique B in G_b. By the same reasoning, B must be a k-partite clique in G, and must be maximal. □

Figures 28 and 29 illustrate the mapping of maximal tricliques to maximal bicliques in a 3-partite set intersection graph. The two partite sets containing genesets in the 3-partite graph in Figure 28 are combined into on partite set in Figure 29. The same method can be applied regardless of the number of partite sets in which vertices represent sets. Therefore enumerating maximal k-partite cliques in a k-partite set intersection graph can be accomplished by enumerating maximal bicliques in a corresponding bipartite graph, for any $k > 2$.

Note that Theorem 4 states that maximal k-partite cliques are actually maximal bicliques when the non-element partite sets are viewed as a single partite set. It does not imply the reverse. That is, maximal bicliques do not necessarily map to maximal k-cliques in such graphs. For instance, vertices {b,c}, b, and c form a maximal biclique in the graph in Figure 28, but not a maximal triclique in the graph in Figure 29.

A corollary to Theorem 4 is that the number of maximal k-partite cliques in an n by m set intersection graph, where n is the number of vertices in the graph excluding the element vertices, and m is the number of element vertices, is bounded by the maximum number bicliques possible in an n by m bipartite graph, $2^a - 2$, where $a = \min(n, m)$.

As a result of Theorem 4, the set of all k-partite cliques in a k-partite set intersection graph are exactly those maximal bicliques in the corresponding bipartite graph that have at least one vertex from each of the partite sets. Stated another way, the problem of enumerating all maximal k-cliques in a k-partite set intersection graph reduces to the problem of enumerating all maximal bicliques in the corresponding bipartite graph, filtering out those maximal bicliques that do not contain vertices from all partite sets.

4.8 Alternate Problem Formulations

With minor changes, the BK-K algorithm can be modified to solve several closely related problems on k-partite graphs, including the following.

1. Find all vertex and/or edge maximum k-partite cliques.
2. Output all cliques with at least (or exactly) k vertices (or edges).
3. Output all cliques with at most k vertices (or edges).

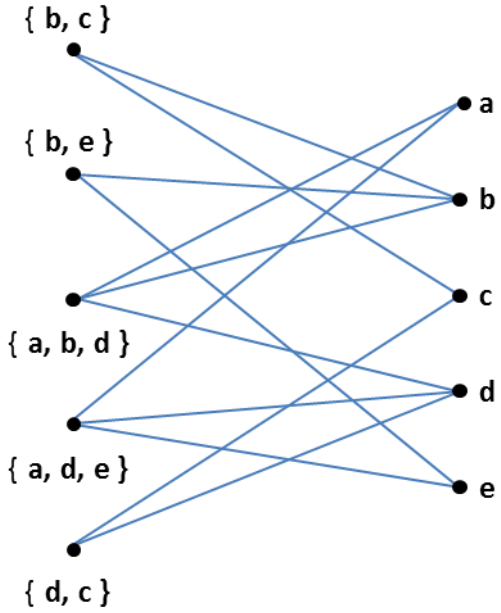

Figure 29. The bipartite graph corresponding to the 3-partite set intersection graph of Figure 28. Vertices representing sets now comprise a single partite set, regardless of how they were partitioned in the 3-partite set intersection graph. Edges between sets have been removed. By Theorem 4, enumerating maximal bicliques in the graph above is will yield all maximal 3-partite cliques in the graph of Figure 28.

With a minor modification to input parsing, the algorithm can be adapted to address the following, more general problem. For purposes of this problem we define a relaxed *k*-partite clique to be *k*-partite clique with any number of additional intrapartite edges. The problem is related to graph coloring.

Input: a graph *G* and a partition of *G* into *k* classes
Output: all relaxed *k*-cliques in *G* with at least one vertex in each of the *k* classes

If we construct graph in the same manner as a *k*-partite set intersection graph, but omit the partite set whose vertices represent set elements, then we obtain a *k*-partite set-set graph. In a *k*-partite set-set graph, the problem of enumerating maximal *k*-partite cliques does not reduce to the problem of enumerating maximal bicliques. This is because in the *k*-partite set intersection graph, all vertices in a *k*-clique must have the same set element in common; in other words, all edges in a *k*-clique represent the same element. In a *k*-partite set-set graph, such is not necessarily the case. Edges in a *k*-clique can represent different set elements.

4.9 Empirical Scaling Tests

We implemented BK-K in C++ and compiled it with g++ under gcc version 4.6.3, using the –O3 flag. Compilation and testing were done on the Ubuntu Linux 12.04 (Precise Pangolin) operating system. All reported timings were obtained on a Dell OptiPlex 9010 Minitower with an Intel Core i7-3770 3.4 GHz processor, 16.0 GB DDR3 non-ECC SDRAM memory at 1600 MHz (4 DIMMs), and a 500 GHz 7200 RPM SATA hard drive.

Since BK-K enumerates maximal k-partite cliques in k-partite graphs for any $k \geq 2$, we first tested it on bipartite graphs, comparing it to MBEA, one of the fastest existing algorithms for enumerating maximal bicliques in bipartite graphs [72]. BK-K is surprisingly competitive, and in fact outperforms MBEA on random graphs when the edge density is above a certain point.

The random graphs we tested were all constructed with partite set sizes selected to yield as close as possible to 4 million potential edges. That is, in each bipartite graph $G = (U, V, E)$ with partite sets U and V and edge set E, $|U| * |V| \approx 4$ million. For instance, the random graph with 2:1 partite set ratio has 2830 and 1415 vertices in its respective partite sets. The number of actual edges varies with density, of course.

The density at which BK-K begins to outperform MBEA depends on the ratio of the partite set sizes. The more unbalanced the sizes, the higher the density at which BK-K overtakes MBEA. See Figures 30 and 31. MBEA is tailored for real-world graphs, which tend to be sparse and are often very unbalanced, with partite set size ratios of 100 or more not being uncommon. So it is not surprising that the relative performance of MBEA improves with lower density and more unbalanced partite sets.

On a series of real-world graphs created from genesets at varying Jaccard similarities, although competitive, BK-K does not quite catch up to the performance of MBEA as the Jaccard similarity threshold decreases. See Figure 32. Unfortunately, unlike with random graphs, it can be difficult to control both density and number of vertices in graphs created from thresholded similarity metrics. Relaxing the threshold can always introduce new edges (except in an already complete graph), but doing so may also introduce new (formerly isolated) vertices. The result can actually be a net decrease in density. The graphs tested for Figure 32 had densities between 0.17 and 0.30, increasing in all but on case as the Jaccard similarity decreased. The largest and highest density graph, created using a Jaccard threshold of 0.13, had more than 313 million maximal bicliques.

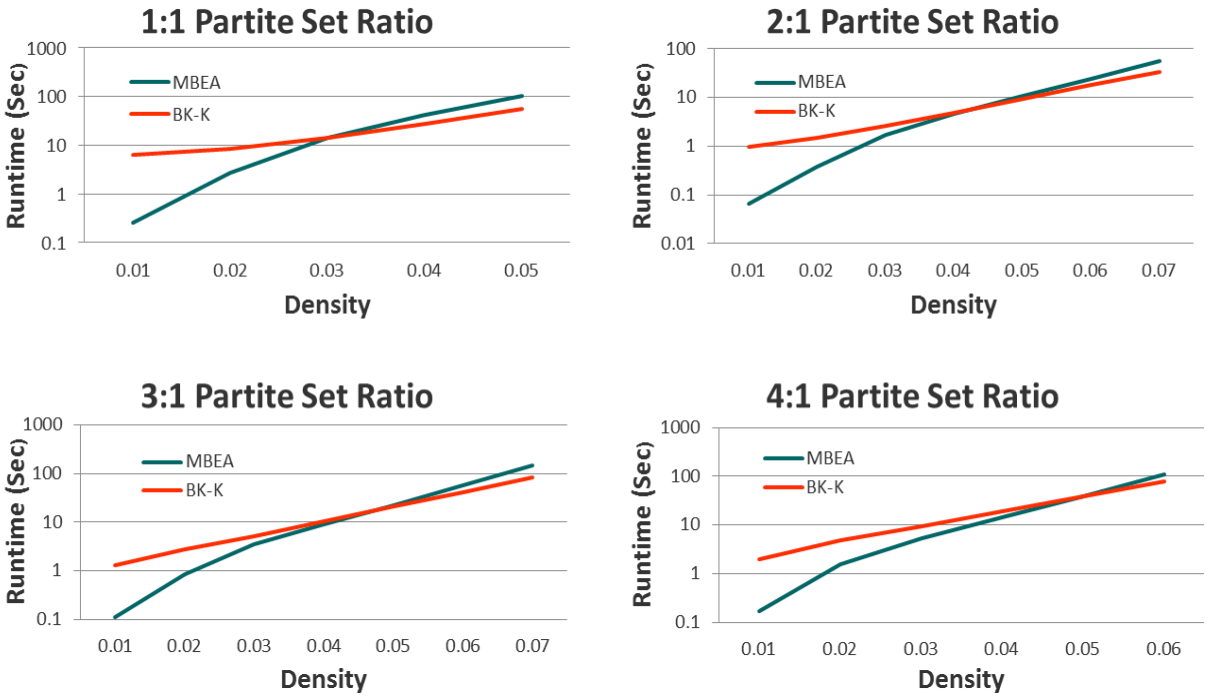

Figure 30. A comparison of MBEA and BK-K on random bipartite graphs with four different partite set size ratios at varying density. MBEA performs better on very sparse graphs, but is overtaken by BK-K as the density increases. As the partite set size ratio increases, the density at which BK-K outperforms MBEA also increases.

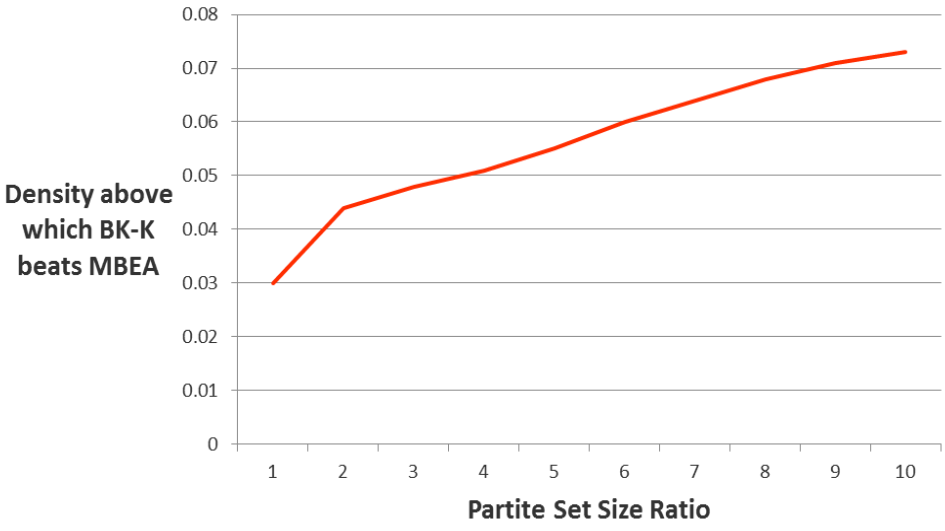

Figure 31. The density above which BK-K outperforms MBEA on random bipartite graphs. The crossover point increases with the ratio of the partite set sizes, on graphs where |U| * |V| ≈ 4,000,000.

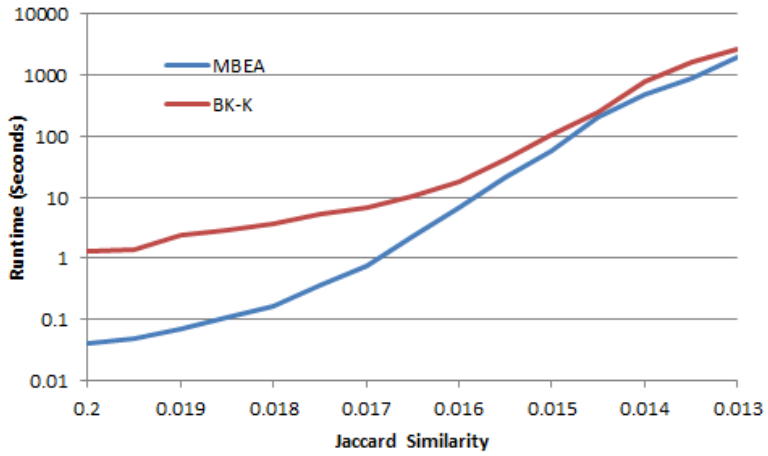

Figure 32. A comparison of MBEA and BK-K on real-world bipartite graphs. The series of graphs was created from a pair of partite sets with varying Jaccard similarity. One partite set has 586 genesets from a GeneWeaver "alcohol" query; the other has 384 genesets mapped to brain regions by the Allen Brain Atlas. The size, density and number of maximal bicliques typically increase as Jaccard similarity decreases.

To investigate how BK-K scales with the size of multipartite graphs, we next tested a suite of random graphs with 3 and 4 partite sets. The graphs were balanced, with partite set sizes from 500 to 7500 vertices and density fixed at 0.01. As shown in Figure 33, when the density is fixed, the number of partite sets makes little difference in the runtime. We also created 5-partite graphs to test, but found that at density of 0.01 the largest 5-partite clique contained only one vertex from each of four partite sets, and two vertices from the fifth, i.e. the largest 5-partite clique was 1x1x1x1x2. This illustrates how increasing the number of partite sets can quickly result in extreme data sparseness, an example of the "curse of dimensionality."

4.10 Preprocessing Heuristics

In this section we present two preprocessing heuristics and describe the results of empirical testing of each. The heuristics are interpartite edge removal and intrapartite edge removal. For the interpartite heuristic, we remove all interpartite edges whose endpoints, which are in different partite sets, have no common neighbor in a third partite set. For the intrapartite heuristic, we remove all intrapartite edges that have no common neighbor in another partite set. When applied prior to adding all intrapartite edges, the interpartite rule results in the removal of all vertices that are not part of a 3-clique. Each heuristic has time

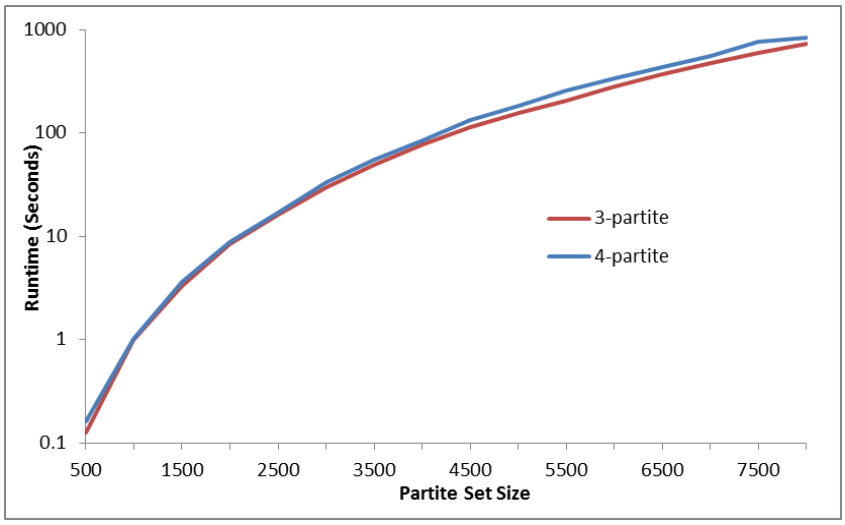

Figure 33. Runtime of BK-K on random balanced 3-partite and 4-partite graphs. The density of the graphs was fixed at 0.01.

complexity $O(n^3)$. The two rules may seem remarkably similar; however empirical testing reveals that the interpartite heuristic works well in practice, especially on graphs with low density, while the intrapartite rule never results in a runtime reduction, and often increases the runtime significantly. See Figure 34 for the effect of each heuristic on random tripartite graphs with 2000 vertices in each partite set.

The interpartite heuristic can be generalized to remove all vertices that are not part of a (k-1)-clique for k-partite graphs where $k > 3$. Given the resulting $O(n^{k-1})$ time complexity, however, it is not clear whether such a generalization will provide any benefit beyond that provided by the basic rule, especially given the sporadic improvement yielded by the basic heuristic.

4.11 Multipartite Summary

We have described the mapping of functional genomics data onto multipartite graphs, and presented an algorithm, BK-K, to enumerate all maximal k-partite cliques in such graphs. We have also discussed an example using graphs derived from keyword-related genesets. Scaling results for an implementation of BK-K suggest that its performance is affected much more by density and the number of maximal k-partite cliques than by the number of partite sets. It even performs well enough on bipartite graphs to be a serious contender against state-of-the-art maximal biclique enumeration algorithm. Multipartite problems arise in the analysis of existing and proposed big data repositories, from which graphs and be constructed and solved by mapping algorithms such as ours onto custom big data

infrastructures. A first impulse when faced with problems of this magnitude, especially in the context of big data, is often to seek an approximate or randomized solution. In contrast, our methods provide the potential for continued scalability with which exact algorithms can be utilized in real-world big data applications.

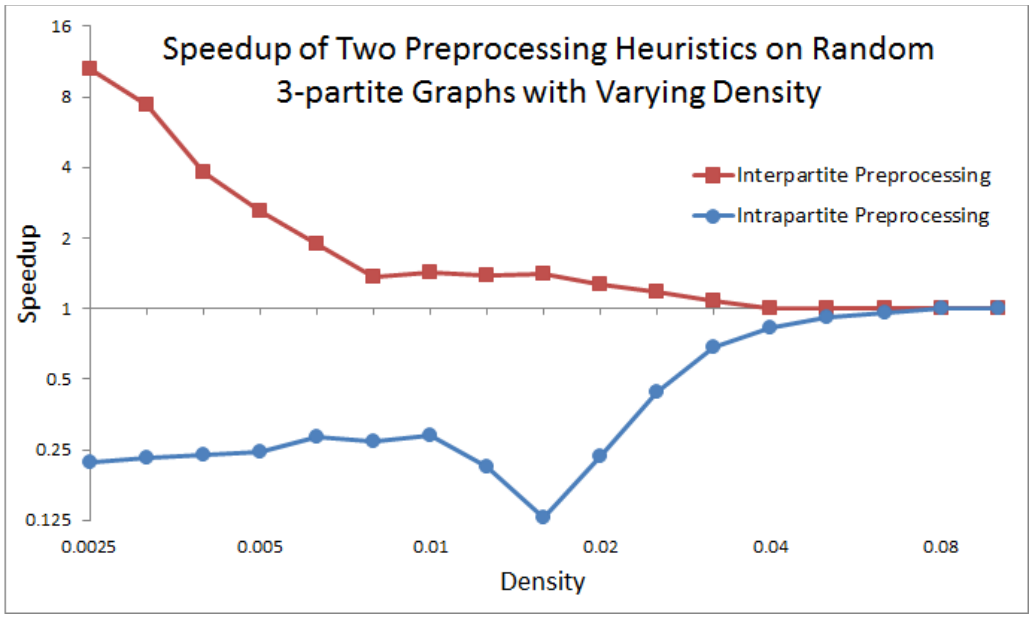

Figure 34. The speedup achieved by interpartite and intrapartite preprocessing. Testing was conducted on random 3-partite graphs with 2000 vertices in each partite at various densities. Interpartite preprocessing is very effective on graphs with low density, being more effective the lower the density. It gradually becomes ineffective as density increases; although at no time does its overhead produce a substantial runtime cost. Conversely, intrapartite preprocessing is never effective. It results in much longer runtimes at low density, and while its overhead eventually becomes insubstantial as density increases, at no point does it provide any benefit.

Chapter 5 Recap and Concluding Remarks

In this work we have discussed the development and application of algorithms for finding dense structure in graphs. The algorithms were applied to a variety of omics data. The algorithms followed a progression from algorithms on general graphs to bipartite graphs to multipartite graphs, and included both algorithmic and theoretical advances, as well as practical implementations and applications. In this chapter we summarize the main contributions of this dissertation and suggest directions for future research.

5.1 Summary of Contributions

In chapter 2, we first described analysis on an acute ethanol response dataset. Contributions included combining gene expression and phenotypes into a single graph in a novel way by using correlations between recombinant inbred mouse strains alongside phenotype measurements about each strain. Genes in several paracliques containing addiction-related phenotypes are candidates for being members of ethanol-responsive regulatory networks. QTL analysis on this dataset further identified potential ethanol-responsive locations on the mouse genome. Of particular interest were QTL's that appeared in multiple paracliques. Such QTL's may indicate signaling hubs between different networks. We next described the analysis of time series data in the developing mouse cerebellum. We developed paraclique signatures to apply to such time series data. Each signature split a paraclique into two anti-correlated parts. Such signatures proved useful for visualizing the behavior of co-expressed groups of genes over time. We then developed an algorithm, based on maximal cliques, to return a preset number of overlapping subgraphs. The algorithm was motivated by a predicted PPI network, for which domain scientists sought a certain number of overlapping protein complexes. The algorithm, however, could be used on any undirected graph where a preset number of overlapping clusters is sought. And finally in chapter 2, we performed experiments on algorithms to enumerate all maximum cliques, demonstrating that the MCE algorithm allowed the solution of transcriptomic graphs that were far beyond the reach of other algorithms.

In chapter 3, we helped improve an algorithm, iMBEA, to enumerate all maximal bicliques. Through empirical experiments on random graphs and genomics graphs, we demonstrated that iMBEA outperforms its closest competitor, namely LCM-MBC, a data mining algorithm. We then extended the notion behind the paraclique algorithm from general graphs to bipartite graphs, resulting in the novel

parabiclique algorithm. In developing the new algorithm, we had to address the difference between a vertex-maximum and edge-maximum biclique. The parabiclique can use either type as the basis for growing parabicliques.

In chapter 4, motivated by the problem of heterogeneous data integration in functional genomics, we investigated maximum and maximal *k*-cliques in multipartite graphs. We were able to prove several new theoretical results. By modifying the Bron-Kerbosch algorithm, we created an algorithm to enumerate all maximal *k*-cliques in a *k*-partite graph, and proved the algorithm was asymptotically optimal. We also showed several alternate problem formulations that the new algorithm could be easily modified to solve.

Although the graph algorithms developed in this dissertation are motivated by particular problems in computational biology, they are designed to work on an abstract graph model, and make no assumptions about the source of their input graphs. Therefore they are widely applicable to data from many diverse fields, as long as the data can be modeled as sets of entities and relationships.

5.2 Future Work

Here we list some of the questions that this research leaves open and suggest avenues for future exploration. We proceed in the order the research was presented.

The gene-phenotype paraclique research in section 2.1 was completed prior to the development of the parabiclique algorithm of section 3.2. Therefore a potential research topic is a comparison of the gene-phenotype paracliques to parabicliques obtained from constructing a bipartite graph with the same data. The cross-paraclique eQTL's in section 2.1 also warrant deeper investigation. Such eQTL's appearance in multiple paracliques may signify hub genes that function as signaling connections between multiple biomolecular networks. It should be possible to validate that a single gene underlies such an eQTL.

The time series signatures of paracliques in section 2.2, which divide the positive and negative correlates into two groups, could be applied to other types of data with a logical ordering of samples, such as dosages, oxygen levels or nutrient levels.

The algorithm designed to find overlapping clusters ("complexes") in section 2.3 is somewhat cumbersome. Further, since its input is the set of all maximal cliques in a graph, it will quickly run into scaling issues when applied to graphs derived from some sources besides PPI networks. It also has a tendency to produce many very large clusters with high overlap. This may be an artifact of either the nature of PPI data or the use of maximal cliques, but in any event a more efficient, scalable

graph-theoretic algorithm that achieves a similar goal (producing a given number of overlapping clusters) can probably be designed.

Our efforts with MCE suggest a number of areas with potential for further investigation. A formal definition of the class of graphs for which ES achieves runtime improvements may lead to new theoretical complexity results, perhaps based upon parameterizing by the amount of maximum clique overlap. Furthermore, such a formal definition may form the basis of a new model for real data graphs. We have noted that the number of disjoint maximum cliques that can be extracted provides an upper bound on the size of an MCC. If we parameterize by the maximum clique size and the number of maximum cliques, does an FPT algorithm exist? In addition, formal mathematical results may be achieved on the sensitivity of the number of maximum cliques to small changes in the graph.

Note that any MCC forms a hitting set over the set of maximum cliques, though not necessarily a minimum one. Also, a set D of disjoint maximum cliques, to which no additional disjoint maximum clique can be added, forms a *subset cover* over the set of all maximum cliques. That is, any maximum clique $C \notin D$ contains at least one $v \in D$. See Figure 35. To the best of our knowledge, this problem has not previously been studied. All we have found in the literature is one citation that erroneously reported it to be one of Karp's original NP-complete problems [131].

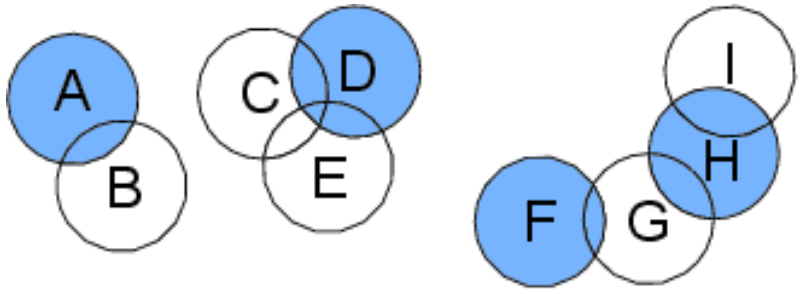

Figure 35. The subset cover problem. The decision version of the subset cover problem asks if there are k or fewer subsets that cover all other subsets. A satisfying solution for k = 4 is the highlighted subsets.

For the subset cover problem, we have noted that it is *NP*-hard by a simple reduction from hitting set. But in the context of MCE we have subsets all of the same size. It may be that this alters the complexity of the problem, or that one can achieve tighter complexity bounds when parameterizing by the subset size. Alternately, consider the problem of finding the minimum subset cover given a known

minimum hitting set. The complexity of this tangential problem is not at all clear, although we conjecture it to be *NP*-complete in and of itself. Lastly, as a practical matter, exploring whether an algorithm that addresses the memory issues of the subset enumeration algorithm presented in [35] and improved in [36] may also prove fruitful. As we have found here, it may well depend at least in part on the data.

The iMBEA algorithm in chapter 3 can possibly be improved. Specifically, an algorithm to enumerate maximal bicliques might be designed that has time complexity $O(2^{n/2})$, which would achieve the asymptotically optimal runtime on bipartite graphs. One could also devise an efficient algorithm to find an edge maximum biclique without needing to enumerate all maximal bicliques in the process. One approach would be to prune the search tree based on the largest biclique found so far, much like current algorithms for finding a maximum clique.

The parabiclique algorithm could be extended to para-*k*-clique algorithm. One drawback of applying such an algorithm, however, may be that as the number of partite sets increases, so does the dimensionality of the data, and thus the sparseness of the graph.

The work on multipartite graphs suggests numerous lines of research that remain to be addressed, both algorithmic and theoretical. For *k*-partite graphs, $k \geq 3$, one cannot do better asymptotic performance than what we present here. However, our results leave open the possibility of a $O(2^{n/2})$ maximal biclique enumeration algorithm. Such an algorithm, or at least its runtime, would be specific to bipartite graphs, since the number of maximal bicliques in general graphs is $3^{n/3}$. Also, it may be possible to do better than our $O(3^{n/3})$ enumeration algorithm if one seeks only to find a vertex-maximum or edge-maximum *k*-partite clique. There also remains the problem of enumerating maximal *k*-partite cliques in general graphs. Theoretical open questions include, for random graphs, both general and *k*-partite, the possibility of deriving an expected number of maximal *k*-partite cliques at a given graph density. In this work we have only considered finding *k*-partite cliques in *k*-partite graphs, but algorithms could be developed to find *k*-partite cliques in general graphs, provided the partition of the graph into partite sets was part of the input.

Applications in biology and other fields may be found for the alternative problem formulations given for the BK-K algorithm. Similarly, all the algorithms and analytic techniques developed in this dissertation, because they operate on abstract graph models, may find applications beyond the data on which they were focused in this dissertation. Such is my hope.

References

1. Setubal JC, Meidanis J: **Introduction to Computational Molecular Biology**. Boston: PWS Publishing Company; 1997.
2. Balasundaram B, Butenko S: **Graph Domination, Coloring and Cliques in Telecommunications**. In: *Handbook of Optimization in Telecommunications.* Edited by Resende MC, Pardalos P: Springer US; 2006: 865-890.
3. Gutiérrez Y, Vázquez S, Montoyo A: **A graph-Based Approach to WSD Using Relevant Semantic Trees and N-Cliques Model**. In: *Computational Linguistics and Intelligent Text Processing.* Edited by Gelbukh A, vol. 7181: Springer Berlin Heidelberg; 2012: 225-237.
4. Pattillo J, Youssef N, Butenko S: **Clique Relaxation Models in Social Network Analysis**. In: *Handbook of Optimization in Complex Networks.* Edited by Thai MT, Pardalos PM: Springer New York; 2012: 143-162.
5. Giarratani F, Hewings G, McCann P: **Handbook of industry studies and economic geography**. Cheltenham, UK ; Northampton, MA: Edward Elgar; 2013.
6. Xu-Hua Y, Bao S, Bo W, You-Xian S: **Mean-field Theory for Some Bus Transport Networks with Random Overlapping Clique Structure**. *Communications in Theoretical Physics* 2010, **53**(4):688.
7. Mehrotra A, Trick MA: **Cliques and clustering: A combinatorial approach**. *Oper Res Lett* 1998, **22**(1):1-12.
8. National Research Council (U.S.). Committee on Mathematical Challenges from Computational Chemistry.: **Mathematical challenges from theoretical/computational chemistry**. Washington, D.C.: National Academy Press; 1995.
9. Bultinck P: **Computational medicinal chemistry for drug discovery**. New York; London: Marcel Dekker ; Momenta; 2004.
10. Stevens K, Kirkpatrick B: **Efficiently Solvable Perfect Phylogeny Problems on Binary and k-State Data with Missing Values**. In: *Algorithms in Bioinformatics.* Edited by Przytycka T, Sagot M-F, vol. 6833: Springer Berlin Heidelberg; 2011: 282-297.
11. Flikkema PG, West B: **Clique-based randomized multiple access for energy-efficient wireless ad hoc networks**. In: *Wireless Communications and Networking, 2003 WCNC 2003 2003 IEEE: 20-20 March 2003 2003*. 977-981 vol.972.
12. Turán P: **On an Extremal Problem in Graph Theory**. *Matematikai és Fizikai Lapok (in Hungarian)* 1941, **48**:436-452.
13. Aigner M: **Turán's Graph Theorem**. *The American Mathematical Monthly* 1995, **102**(9):808-816.

14. West DB: **Introduction to graph theory**, 2nd edn. Upper Saddle River, N.J.: Prentice Hall; 2001.
15. Perkins AD, Langston MA: **Threshold Selection in Gene Co-Expression Networks Using Spectral Graph Theory Techniques**. *BMC Bioinformatics* 2009, **10**.
16. Zhang B, Horvath S: **A general framework for weighted gene co-expression network analysis**. *Statistical Applications in Genetics and Molecular Biology* 2005, **4**(1):article 17.
17. Borate BR, Chesler EJ, Langston MA, Saxton AM, Voy BH: **Comparison of threshold selection methods for microarray gene co-expression matrices**. *BMC Research Notes* 2009, **2**.
18. Palla G, Derényi I, Farkas I, Vicsek T: **Uncovering the overlapping community structure of complex networks in nature and society**. *Nature* 2005, **435**(7043):814-818.
19. Chesler EJ, Langston MA: **Combinatorial genetic regulatory network analysis tools for high throughput transcriptomic data**. In: *Systems Biology and Regulatory Genomics.* Edited by Eskin E, vol. 4023: Springer; 2006: 150–165.
20. Chesler EJ, Lu L, Shou S, Qu Y, Gu J, Wang J, Hsu HC, Mountz JD, Baldwin NE, Langston MA *et al*: **Complex trait analysis of gene expression uncovers polygenic and pleiotropic networks that modulate nervous system function**. *Nature Genetics* 2005, **37**(3):233-242.
21. Jay J, Eblen J, Zhang Y, Benson M, Perkins A, Saxton A, Voy B, Chesler E, Langston M: **A systematic comparison of genome-scale clustering algorithms**. *BMC Bioinformatics* 2012, **13**(Suppl 10):S7.
22. McClosky B, Hicks I: **Combinatorial algorithms for the maximum k-plex problem**. *J Comb Optim* 2012, **23**(1):29-49.
23. Karp R: **Reducibility among combinatorial problems**. In: *Complexity of Computer Computations.* Edited by Miller R, Thatcher J: Plenum Press; 1972: 85--103.
24. Downey RG, Fellows MR: **Parameterized Complexity**. New York: Springer; 1999.
25. Hastad J: **Clique is Hard to Approximate Within $n^{1-\varepsilon}$**. In: *Proceedings of the 37th Annual Symposium on Foundations of Computer Science 1996*. IEEE Computer Society.
26. Garey MR, Johnson DS: **Computers and Intractability: A Guide to the Theory of NP-Completeness**: W. H. Freeman and Company; 1979.
27. Abu-Khzam FN, Langston MA, Shanbhag P, Symons CT: **Scalable parallel algorithms for FPT problems**. *Algorithmica* 2006, **45**(3):269-284.

28. Langston MA, Perkins AD, Saxton AM, Scharff JA, Voy BH: **Innovative Computational Methods for Transcriptomic Data Analysis: A Case Study in the Use of FPT for Practical Algorithm Design and Implementation**. *The Computer Journal* 2008, **51**:26-38.
29. Niedermeier R: **Invitation to Fixed-Parameter Algorithms**: Oxford University Press; 2006.
30. Eblen JD: **The Maximum Clique Problem: Algorithms, Applications, and Implementations**. *PhD Dissertation.* University of Tennessee; 2010.
31. Eblen JD, Gerling IC, Saxton AM, Wu J, Snoddy JR, Langston MA: **Graph Algorithms for Integrated Biological Analysis, with Applications to Type 1 Diabetes Data**. In: *Clustering Challenges in Biological Networks.* Edited by Chaovalitwongse WA: World Scientific; 2009: 207-222.
32. Baldwin NE, Collins RL, Langston MA, Leuze MR, Symons CT, Voy. BH: **High Performance Computational Tools for Motif Discovery**. In: *Proceedings, IEEE International Workshop on High Performance Computational Biology (HiCOMB): 2004; Santa Fe, New Mexico*.
33. Fernández-Baca D: **The Perfect Phylogeny Problem**. In: *Steiner Trees in Industry.* Edited by Cheng X, Du D-Z: Springer; 2002.
34. Bomze I, Budinich M, Pardalos P, Pelillo M: **The Maximum Clique Problem**. In: *Handbook of Combinatorial Optimization.* Edited by Du D-Z, Pardalos PM, vol. 4: Kluwer Academic Publishers; 1999.
35. Kose F, Weckwerth W, Linke T, Fiehn O: **Visualizing plant metabolomic correlation networks using clique–metabolite matrices**. *Bioinformatics* 2001, **17**:1198-1208.
36. Zhang Y, Abu-Khzam FN, Baldwin NE, Chesler EJ, Langston MA, Samatova NF: **Genome-scale computational approaches to memory-intensive applications in systems biology**. In: *Proceedings, Supercomputing: 2005; Seattle, Washington*.
37. Bron C, Kerbosch J: **Algorithm 457: finding all cliques of an undirected graph**. *Proceedings of the ACM* 1973, **16(9)**:575-577.
38. Harley ER: **Comparison of Clique-Listing Algorithms**. In: *International Conference on Modeling, Simulation and Visualization Methods: 2004 2004; Las Vegas*. CSREA Press: 433-438.
39. Moon JW, Moser. L: **On Cliques in Graphs**. *Israel J Math* 1965, **3**:23-28.
40. Tomita E, Tanaka A, Takahashi H: **The Worst-Case Time Complexity for Generating all Maximal Cliques and Computational Experiments**. *Theoretical Computer Science* 2006, **363**:28-42.

41. Wolen AR, Phillips CA, Langston MA, Putman AH, Vorster PJ, Bruce NA, York TP, Williams RW, Miles MF: **Genetic dissection of acute ethanol responsive gene networks in prefrontal cortex: functional and mechanistic implications**. *PLoS One* 2012, **7**(4):e33575.
42. Zhang L, Wang L, Ravindranathan A, Miles MF: **A new algorithm for analysis of oligonucleotide arrays: application to expression profiling in mouse brain regions**. *Journal of molecular biology* 2002, **317**(2):225-235.
43. R. E. Kennedy, R. T. Kerns, X. Kong, K. J. Archer, Miles MF: **SScore: an R package for detecting differential gene expression without gene expression summaries.** *Bioinformatics* 2006, **22**(10):1272-1274.
44. Fisher RA: **Statistical methods for research workers**. Edinburgh, London,: Oliver and Boyd; 1925.
45. Kugler KG, Mueller LA, Graber A: **MADAM - An open source meta-analysis toolbox for R and Bioconductor**. *Source code for biology and medicine* 2010, **5**:3.
46. Storey JD, Tibshirani R: **Statistical significance for genomewide studies**. *Proceedings of the National Academy of Sciences of the United States of America* 2003, **100**(16):9440-9445.
47. Csardi G, Nepusz T: **The igraph Software Package for Complex Network Research**. *InterJournal* 2006, **Complex Systems**:1695.
48. Chen J, Bardes EE, Aronow BJ, Jegga AG: **ToppGene Suite for gene list enrichment analysis and candidate gene prioritization**. *Nucleic Acids Res* 2009, **37**(Web Server issue):W305-311.
49. Williams RW, Gu J, Qi S, Lu L: **The Genetic Structure of Recombinant Inbred Mice: High-Resolution Consensus Maps for Complex Trait Analysis**. *Genome Biology* 2001, **2**:0046.0041-0046.0018.
50. Shifman S, Bell JT, Copley RR, Taylor MS, Williams RW, Mott R, Flint J: **A high-resolution single nucleotide polymorphism genetic map of the mouse genome**. *PLoS biology* 2006, **4**(12):e395.
51. Ha T, Swanson D, Larouche M, Glenn R, Weeden D, Zhang P, Hamre K, Langston M, Phillips C, Song M *et al*: **CbGRiTS: Cerebellar gene regulation in time and space**. *Developmental Biology* 2015, **397**(1):18-30.
52. Shumway RH, Stoffer DS: **Time Series Analysis and Its Applications (Springer Texts in Statistics)**, 3 edn: Springer-Verlag New York, Inc.; 2011.
53. Voy BH, Scharff JA, Perkins AD, Saxton AM, Borate B, Chesler EJ, Branstetter LK, Langston MA: **Extracting gene networks for low dose radiation using graph theoretical algorithms**. *PLoS Computational Biology* 2006, **2**(7):e89.

54. Huang DW, Sherman BT, Lempicki RA: **Systematic and integrative analysis of large gene lists using DAVID Bioinformatics Resources**. *Nature Protocols* 2009, **4**(1):44-57.
55. Huang DW, Sherman BT, Lempicki RA: **Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists**. *Nucleic Acids Research* 2009, **37**(1):1-13.
56. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT *et al*: **Gene ontology: tool for the unification of biology**. *Nature Genetics* 2000, **25**:25–29.
57. Schoenrock A, Samanfar B, Pitre S, Hooshyar M, Jin K, Phillips CA, Wang H, Phanse S, Omidi K, Gui Y *et al*: **Efficient prediction of human protein-protein interactions at a global scale**. *BMC Bioinformatics* 2014, **15**(1):383.
58. Khan SH, Ahmad F, Ahmad N, Flynn DC, Kumar R: **Protein-protein interactions: principles, techniques, and their potential role in new drug development**. *Journal of biomolecular structure & dynamics* 2011, **28**(6):929-938.
59. Nibbe RK, Chowdhury SA, Koyuturk M, Ewing R, Chance MR: **Protein-protein interaction networks and subnetworks in the biology of disease**. *Wiley interdisciplinary reviews Systems biology and medicine* 2011, **3**(3):357-367.
60. Eblen JD, Phillips CA, Rogers GL, Langston MA: **The maximum clique enumeration problem: algorithms, applications, and implementations**. *BMC Bioinformatics* 2012, **13 Suppl 10**:S5.
61. Baldwin NE, Chesler EJ, Kirov S, Langston MA, Snoddy JR, Williams RW, Zhang B: **Computational, integrative and comparative methods for the elucidation of genetic co-expression networks**. *Journal of Biomedicine and Biotechnology* 2005, **2**:172-180.
62. Rota Bulò S, Torsello A, Pelillo M: **A game–theoretic approach to partial clique enumeration**. *Image Vision Comput* 2009, **27**(7):911-922.
63. Fernau H: **On Parameterized Enumeration**. In: *Computing and Combinatorics.* Edited by Ibarra O, Zhang L, vol. 2387: Springer Berlin Heidelberg; 2002: 564-573.
64. Rogers GL, Perkins AD, Phillips CA, Eblen JD, Abu-Khzam FN, Langston MA: **Using out-of-core techniques to produce exact solutions to the maximum clique problem on extremely large graphs**. In: *Proceedings, ACS/IEEE International Conference on Computer Systems and Applications: 2009; Rabat, Morocco*. 374-381.
65. Tomita E, Kameda T: **An Efficient Branch-and-bound Algorithm for Finding a Maximum Clique with Computational Experiments**. *J Glob Optim* 2007, **37**(1):95-111.

66. Edgar R, Domrachev M, Lash AE: **Gene Expression Omnibus: NCBI gene expression and hybridization array data repository**. *Nucleic Acids Res* 2002, **30**(1):207-210.
67. Erdös P, Rényi A: **On the evolution of random graphs**. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences* 1960, **5**:17-61.
68. Barabási AL, Albert R: **Emergence of scaling in random networks**. *Science* 1999, **286**:509-512.
69. Li L, Alderson D, Doyle JC, Willinger W: **Towards a Theory of Scale Free Graphs: Definition, Properties, and Implications**. *Internet Mathematics* 2005, **2**(4):431-523.
70. Keller EF: **Revisiting "scale-free" networks**. *BioEssays : news and reviews in molecular, cellular and developmental biology* 2005, **27**(10):1060-1068.
71. Zhang Y: **Scalable Graph Algorithms with Applications in Genetics**. University of Tennessee; 2008.
72. Zhang Y, Phillips CA, Rogers GL, Baker EJ, Chesler EJ, Langston MA: **On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types**. *BMC Bioinformatics* 2014, **15**(1):110.
73. Malgrange Y: **Recherche des sous-matrices premières d'une matrice à coefficients binaires. Applications à certains problèmes de graphe**. In: *Deuxième Congrès de l'AFCALTI: 1962; Paris*. Gauthier-Villars.
74. Berry A, Bordat J-P, Sigayret A: **A local approach to concept generation**. *Ann Math Artif Intell* 2007, **49**(1-4):117-136.
75. Kuznetsov SO, Obiedkov S: **Comparing Performance of Algorithms for Generating Concept Lattices**. *Journal of Experimental and Theoretical Artificial Intelligence* 2002, **14**:189-216.
76. Kaytoue-Uberall M, Duplessis S, Napoli A: **Using Formal Concept Analysis for the Extraction of Groups of Co-expressed Genes**. In: *Modelling, Computation and Optimization in Information Systems and Management Sciences.* Edited by Le Thi H, Bouvry P, Pham Dinh T, vol. 14: Springer Berlin Heidelberg; 2008: 439-449.
77. Kaytoue M, Kuznetsov SO, Napoli A, Duplessis S: **Mining gene expression data with pattern structures in formal concept analysis.** *Journal of Information Sciences: Special Issue on Information Engineering Applications Based on Lattices* 2011, **181**(10):1989-2001.
78. Cheng Y, Church GM: **Biclustering of expression data**. In: *Proceedings, International Conference on Intelligent Systems for Molecular Biology: 2000*. 93-103.

79. Tanay A, Sharan R, Shamir R: **Discovering Statistically Significant Biclusters in Gene Expression Data**. *Bioinformatics* 2002, **18**:136-144.
80. Wang H, Wang W, Yang J, Yu PS: **Clustering by pattern similarity in large data sets**. In: *Proceedings of the 2002 ACM SIGMOD international conference on Management of data; Madison, Wisconsin*. 564737: ACM 2002: 394-405.
81. Sanderson MJ, Driskell AC, Ree RH, Eulenstein O, Langley S: **Obtaining maximal concatenated phylogenetic data sets from large sequence databases**. *Mol Biol Evol* 2003, **20**(7):1036-1042.
82. Baker EJ, Jay JJ, Philip VM, Zhang Y, Li Z, Kirova R, Langston MA, Chesler EJ: **Ontological Discovery Environment: a system for integrating gene-phenotype associations**. *Genomics* 2009, **94**(6):377-387.
83. Kirova R, Langston MA, Peng X, Perkins AD, Chesler EJ: **A systems genetic analysis of chronic fatigue syndrome: combinatorial data integration from SNPs to differential diagnosis of disease**. In: *Methods of Micorarray Data Analysis VI.* Edited by McConnell P, Lim S, Cuticchia AJ. Scotts Valley, California: CreateSpace Publishing; 2009: 81-98.
84. Mushlin RA, Kershenbaum A, Gallagher ST, Rebbeck TR: **A graph-theoretical approach for pattern discovery in epidemiological research**. *IBM Systems Journal* 2007, **46**(1):135-149.
85. Liu J, Wang W: **OP-Cluster: Clustering by Tendency in High Dimensional Space**. In: *Proceedings of the Third IEEE International Conference on Data Mining*. 952138: IEEE Computer Society 2003: 187.
86. Baker EJ, Jay JJ, Bubier JA, Langston MA, Chesler EJ: **GeneWeaver: a web-based system for integrative functional genomics**. *Nucleic Acids Res* 2012, **40**(Database issue):D1067-1076.
87. Peeters R: **The maximum edge biclique problem is NP-complete**. *Discrete Applied Mathematics* 2003, **131**(3):651-654.
88. Eppstein D: **Arboricity and bipartite subgraph listing algorithms**. *Inf Process Lett* 1994, **51**(4):207-211.
89. Makino K, Uno T: **New Algorithms for Enumerating All Maximal Cliques**. In: *Algorithm Theory - SWAT 2004.* Edited by Hagerup T, Katajainen J, vol. 3111: Springer Berlin Heidelberg; 2004: 260-272.
90. Zaki MJ, Ogihara M: **Theoretical foundations of association rules** In: *3rd ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery: 1998; Seattle, Washington*.
91. Li J, Li H, Soh D, Wong L: **A Correspondence Between Maximal Complete Bipartite Subgraphs and Closed Patterns**. In: *Knowledge Discovery in*

Databases: PKDD 2005. Edited by Jorge A, Torgo L, Brazdil P, Camacho R, Gama J, vol. 3721: Springer Berlin Heidelberg; 2005: 146-156.

92. Zaki MJ, Hsiao C-j: **CHARM: an efficient algorithm for closed itemset mining**. In: *Proceedings of the 2002 SIAM international conference on data mining: 2002; Arlington, VA*. 457--473.
93. Wang J, Han J, Pei J: **CLOSET+: searching for the best strategies for mining frequent closed itemsets**. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining; Washington, D.C.* 956779: ACM 2003: 236-245.
94. Grahne G, Zhu J: **Efficiently using prefix-trees in mining frequent itemsets**. In: *IEEE Workshop on Frequent Itemset Mining Implementations: 2003; Melbourne, FL*.
95. Grahne G, Zhu J: **Reducing the main memory consumptions of FPmax* and FPclose**. In: *IEEE Workshop on Frequent Itemset Mining Implementations: 2004; Brighton, UK*.
96. Uno T, Kiyomi M, Arimura H: **LCM ver.2: Efficient mining algorithms for frequent/closed/maximal itemsets**. In: *IEEE Workshop on Frequent Itemset Mining Implementations: 2004; Brighton, UK*.
97. Li J, Liu G, Li H, Wong L: **Maximal Biclique Subgraphs and Closed Pattern Pairs of the Adjacency Matrix: A One-to-One Correspondence and Mining Algorithms**. *IEEE Trans on Knowl and Data Eng* 2007, **19**(12):1625-1637.
98. Alexe G, Alexe S, Crama Y, Foldes S, Hammer PL, Simeone B: **Consensus algorithms for the generation of all maximal bicliques**. *Discrete Appl Math* 2004, **145**(1):11-21.
99. Liu G, Sim K, Li J: **Efficient mining of large maximal bicliques**. In: *Proceedings of the 8th international conference on Data Warehousing and Knowledge Discovery; Krakow, Poland*. Springer-Verlag 2006: 437-448.
100. Chesler EJ, Wang J, Lu L, Qu Y, Manly KF, Williams RW: **Genetic Correlates of Gene Expression in Recombinant Inbred Strains: a Relational Model System to Explore Neurobehavioral Phenotypes**. *Neuroinformatics* 2003, **1(4)**:343-357.
101. Phillips CA, Jay JJ, Baker EJ, Chesler EJ, Langston MA: **On bipartite graph decomposition in the presence of noise, with applications to biological data clustering**. In: *11th Cologne-Twente Workshop on Graphs and Combinatorial Optimization: 2012; Munich, Germany*. 215-219.
102. Pati A, Vasquez-Robinet C, Heath LS, Grene R, Murali TM: **XcisClique: analysis of regulatory bicliques**. *BMC Bioinformatics* 2006, **7**:218.

103. Schweiger R, Linial M, Linial N: **Generative probabilistic models for protein-protein interaction networks--the biclique perspective**. *Bioinformatics* 2011, **27**(13):i142-148.
104. Fan Z-J, Liao M-X, He X-X, Hu X-H, Zhou X: **Efficient algorithm for extreme maximal biclique mining in cognitive frequency decision making**. In: *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on: 27-29 May 2011 2011*. 25-30.
105. Madeira SC, Oliveira AL: **Biclustering algorithms for biological data analysis: a survey**. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2004, **1**:24-45.
106. Tanay A, Sharan R, Shamir R: **Biclustering Algorithms: A Survey**. In: *Handbook of Bioinformatics.* 2004.
107. Phillips CA, Wang K, Bubier J, Baker EJ, Chesler EJ, Langston MA: **Scalable multipartite subgraph enumeration for integrative analysis of heterogeneous experimental functional genomics data**. In: *ACM International Workshop on Big Data in Life Sciences: 2015*.
108. Jay JJ: **Cross Species Integration of Functional Genomics Experiments**. *International Review of Neurobiology* 2012, **104**:1-24.
109. Hagan RD, Phillips CA, Wang K, Rogers GL, Langston MA: **Toward an efficient, highly scalable maximum clique solver for massive graphs**. In: *IEEE International Conference on Big Data: 27-30 Oct. 2014 2014*. 41-45.
110. Miller RE, Muller DE: **A problem of maximum consistent subsets.** *IBM Research Report RC-240, Watson Research Center, Yorktown Heights, NY* 1960.
111. Wood D: **On the Number of Maximal Independent Sets in a Graph**. *Discrete Mathematics & Theoretical Computer Science* 2011, **13**:17-20.
112. Dean J, Ghemawat S: **MapReduce: simplified data processing on large clusters**. *Commun ACM* 2008, **51**(1):107-113.
113. White T: **Hadoop: The Definitive Guide**: O'Reilly Media, Inc.; 2009.
114. Pardalos P, Vavasis S: **Quadratic programming with one negative eigenvalue is NP-hard**. *J Glob Optim* 1991, **1**(1):15-22.
115. Clementi AF, Crescenzi P, Rossi G: **On the Complexity of Approximating Colored-Graph Problems Extended Abstract**. In: *Computing and Combinatorics.* Edited by Asano T, Imai H, Lee DT, Nakano S-i, Tokuyama T, vol. 1627: Springer Berlin Heidelberg; 1999: 281-290.
116. Davis AP, Grondin CJ, Lennon-Hopkins K, Saraceni-Richards C, Sciaky D, King BL, Wiegers TC, Mattingly CJ: **The Comparative Toxicogenomics Database's 10th year anniversary: update 2015**. *Nucleic Acids Res* 2015, **43**(Database issue):D914-920.

117. Castro VM, Minnier J, Murphy SN, Kohane I, Churchill SE, Gainer V, Cai T, Hoffnagle AG, Dai Y, Block S *et al*: **Validation of Electronic Health Record Phenotyping of Bipolar Disorder Cases and Controls**. *American Journal of Psychiatry* 2015, **172**(4).
118. Potash JB: **Electronic Medical Records: Fast Track to Big Data in Bipolar Disorder**. *The American Journal of Psychiatry* 2015.
119. Torrente MP, Freeman WM, Vrana KE: **Protein biomarkers of alcohol abuse**. *Expert Review of Proteomics* 2012, **9**(4):425-436.
120. Clinton SM, Stead JDH, Miller S, Watson SJ, Akil H: **Developmental underpinnings of differences in rodent novelty-seeking an emotional reactivity**. *The European Journal of Neuroscience* 2011, **34**(6):994-1005.
121. Cui C, Shurtleff D, Harris RA: **Neuroimmune Mechanisms of Alcohol and Drug Addiction**. *International Review of Neurobiology* 2014, **118**:1-12.
122. Mayfield J, Ferguson L, Harris RA: **Neuroimmune Signaling: A Key Component of Alcohol Abuse**. *Current opinion in neurobiology* 2013, **23**(4):513-520.
123. Jones KA, Thomsen C: **The Role of the Innate Immune System in Psychiatric Disorders**. *Molecular and Cellular Neuroscience* 2013, **53**:52-62.
124. Miller AH, Haroon E, Raison CL, Felger JC: **Cytokine Targets in the Brain: Impact on Neurotransmitters and Neurocircuits**. *Depression and anxiety* 2013, **30**(4):297-306.
125. Gaspers S, Kratsch D, Liedloff M: **On Independent Sets and Bicliques in Graphs**. *Algorithmica* 2012, **62**(3-4):637-658.
126. Eppstein D, Löffler M, Strash D: **Listing All Maximal Cliques in Sparse Graphs in Near-Optimal Time**. In: *Algorithms and Computation*. Edited by Cheong O, Chwa K-Y, Park K, vol. 6506: Springer Berlin Heidelberg; 2010: 403-414.
127. Zhang Y, Abu-Khzam FN, Baldwin NE, Chesler EJ, Langston MA, Samatova NF: **Genome-Scale Computational Approaches to Memory-Intensive Applications in Systems Biology**. In: *Supercomputing, 2005 Proceedings of the ACM/IEEE SC 2005 Conference: 12-18 Nov. 2005 2005*. 12-12.
128. Abu-Khzam FN, Baldwin NE, Langston MA, Samatova NF: **On the Relative Efficiency of Maximal Clique Enumeration Algorithms, with Application to High-Throughput Computational Biology**. In: *Proceedings, International Conference on Research Trends in Science and Technology: 2005; Beirut, Lebanon*.
129. Zaki MJ, Peters M, Assent I, Seidl T: **Clicks: An effective algorithm for mining subspace clusters in categorical datasets**. *Data & Knowledge Engineering* 2007, **60**(1):51-70.

130. Liu Q, Chen Y-PP, Li J: **k-Partite cliques of protein interactions: A novel subgraph topology for functional coherence analysis on PPI networks**. *Journal of Theoretical Biology* 2014, **340**(0):146-154.
131. Malouf R: **Maximal Consistent Subsets**. *Comput Linguist* 2007, **33**(2):153-160.

Vita

Charles Phillips was born in Columbia, Missouri. After graduating from Rock Bridge High School, he attended the University of Missouri for two years, majoring first in Physics and then in Computer Science. He left school to pursue a career in restaurant management, working his way up to General Manager at Domino's Pizza. After working in various capacities at Godfather's Pizza and Papa John's Pizza, he returned to school. He earned an Associate of Applied Science degree in Computer Information Systems from Moberly Area Community College in 2003 and went on to earn a Bachelor of Science with Distinction degree in Computer Science from Columbia College of Missouri in 2005. He is currently a PhD candidate at the University of Tennessee. During his graduate studies he worked as a research assistant in the Biosciences Division at Oak Ridge National Laboratory. While at the University of Tennessee he has received the Chancellor's Citation for Extraordinary Professional Promise and was named the ACM/IEEE Computer Science Teaching Assistant of the Year. He is currently studying under the direction of Dr. Michael A. Langston.