8-2015

Analytical, Theoretical and Empirical Advances in Genome-Scale Algorithmics

Kai Wang
University of Tennessee - Knoxville, kwang11@vols.utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Kai Wang entitled "Analytical, Theoretical and Empirical Advances in Genome-Scale Algorithmics." 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:

Jian Huang, Michael A. Langston, Bruce MacLennan, Xiaoyan Zhu

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

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

Analytical, Theoretical and Empirical Advances in Genome-Scale Algorithmics

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Kai Wang

August 2015

To my parents.

Acknowledgements

I would like to thank my advisor Dr. Michael A. Langston for his continuous support, advice, inspiration, encouragement and help over my PhD study here. I appreciate my committee members Dr. Bruce MacLennan, Dr. Jian Huang, Dr. Lynne E. Parker and Dr. Xiaoyan Zhu for their time reading my dissertation and giving suggestions to improve it. Dr. Arnold M. Saxton taught me valuable knowledge and skills of statistical analysis. My colleagues Charles A. Phillips, Ronald D. Hagan, Allan Yuping Lu and Gary L. Rogers helped me through discussions about numerous research and technical problems. I would like to express my sincere gratitude to my former advisors Dr. Xukai Zou and Dr. Yuanshun Dai for their valuable advice and support. I would not have the opportunity for my overseas studies if it were not for Dr. Zengke Zhang, Dr. Chundi Mu and Dr. Geng Yang, who believed in me and wrote recommendation letters for me. I would also like to thank many other professors at the University of Tennessee, IUPUI and Tsinghua University who taught me valuable knowledge and/or wrote recommendation letters for me. Finally, my past teachers in elementary school, middle school and high schools enlightened me about the natural beauty of the world.

The only reason for time is so that everything doesn't happen at once.

——— *Albert Einstein*

Abstract

Ever-increasing amounts of complex biological data continue to come on line daily. Examples include proteomic, transcriptomic, genomic and metabolomic data generated by a plethora of high-throughput methods. Accordingly, fast and effective data processing techniques are more and more in demand. This issue is addressed in this dissertation through an investigation of various algorithmic alternatives and enhancements to routine and traditional procedures in common use. In the analysis of gene co-expression data, for example, differential measures of entropy and variation are studied as augmentations to mere differential expression. These novel metrics are shown to help elucidate disease-related genes in wide assortments of case/control data. In a more theoretical spirit, limits on the worst-case behavior of density based clustering methods are studied. It is proved, for instance, that the well-known paraclique algorithm, under proper tuning, can be guaranteed never to produce subgraphs with density less than 2/3. Transformational approaches to efficient algorithm design are also considered. Classic graph search problems are mapped to and from well-studied versions of satisfiability and integer linear programming. In so doing, regions of the input space are classified for which such transforms are effective alternatives to direct graph optimizations. In all these efforts, practical implementations are emphasized in order to advance the boundary of effective computation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the rapid development of computational biology and bioinformatics, ever-increasing amounts of complex biological data continue to come on line daily. Among these are proteomic, transcriptomic, genomic and metabolomic data generated by a plethora of high-throughput methods in labs around the world. Accordingly, fast and effective data processing techniques are to a greater extent in demand each day. This dissertation investigates several algorithmic alternatives and improvements to routine and traditional procedures of high-throughput biological data analysis methods in common use. The main contributions are divided and discussed in the following three chapters. Chapter 2 introduces two analytical differential metrics for case/control biological data analysis. Chapter 3 gives theoretical analysis of performance guarantee of a clustering method. Chapter 4 conducts an empirical study of performance comparisons of several graph optimization methods. In this chapter we review some common background and motivation for these chosen research topics and define the terminologies and notations used throughout this dissertation.

1.1 Background

1.1.1 A Brief History of Computational Biology

One of the main application areas of the methods we investigate in this dissertation is computational biology, a field with increasingly supporting role in the medical and pharmaceutical sciences. Advances in these scientific fields could have huge and everlasting impacts for future generations of the human society. In the following we briefly review the provenance of the field of computational biology.

From 1856 to 1863 Gregor Mendel investigated inheritable traits through pea plant experiments and coined the terms *dominant* and *recessive* traits. Many of the rules of heredity he established are now known as the laws of *Mendelian inheritance*. In 1902 Walter Sutton created the term *gene* to describe genetic factors located on chromosomes and developed the chromosomal theory of heredity, which was later elaborated and expanded by Thomas Hunt Morgan and his students. In 1952 Rosalind Franklin and Maurice Wilkins conducted X-ray crystallography studies of DNA, which led to the elucidation of the structure of DNA. In 1953 James Watson and Francis Crick proposed the double-stranded, helical model for DNA, which won them Nobel prize in Physiology or Medicine in 1962. In 1956 Francis Crick and George Gamov introduced the Central Dogma of Molecular Biology to explain protein synthesis from DNA. In 1986 Applied Biosystems introduced the first automated DNA fluorescence sequencer and since then, the complete genome of a few organisms (e.g, yeast and E. coli) were sequenced. In 1990 the Human Genome Project was launched and it was completed in 2003.

The inception of the companies like Affymetrix and Illumina has greatly accelerated the sequencing and microarray based solutions for gene expression measurement. When a gene is used to produce mRNA molecules and then proteins, it is said to be *expressed*. Gene Expression Omnibus [28] has become one of the central repositories for high-throughput gene expression data. A common type of gene expression data is

the *case/control* data, where *case* means the data come from subjects with symptoms under certain treatment and *control* means the data come from subjects without symptoms. Investigations based on case/control data are common for research of drug development. We note in passing that the results of Chapter 1 are derived from research based on such gene expression data that led to ontological discoveries.

1.1.2 Review of Graph Theory and Computational Complexity: Concepts and Nomenclature

The topics of chapters 3 and 4 are related to graph algorithms and/or computational complexity theory with their applications. In this section we briefly review the basic concepts and relevant notations for these fields. They can be found in any standard textbook about graph theory and computational complexity.

All graphs discussed in this dissertation are finite, simple, unweighted, and undirected unless stated otherwise. A graph $G = (V, E)$ consists of a set of vertices V and a set of edges E, where each edge $e \in E$ is a two-element subset of V. If $\{u, v\} \in E$, then the vertices u and v are said to be *adjacent*, otherwise they are *nonadjacent*. The set of vertices adjacent to a vertex u is called the *neighborhood* of u and denoted $N(u)$. The *degree* of a vertex is the size of its neighborhood. A *subgraph* of $G = (V, E)$ is a graph $G' = (V', E')$, where $V' \subseteq V$, $E' \subseteq E$. A subgraph $G' = (V', E')$ of $G = (V, E)$ is said to be an *induced* subgraph if $V' \subseteq V$ and $E' = \{\{u, v\} \in E | u \in V', v \in V'\}$. A *clique* of a graph $G = (V, E)$ is a subset C of V such that $u, v \in C \Rightarrow \{u, v\} \in E$. A *maximum* clique of a graph G is a clique whose size is no less than any other clique of G. A *maximal* clique of a graph G is a clique that is not contained in any other clique. An independent set I of a graph $G = (V, E)$ is a subset I of V such that $u, v \in I \Rightarrow \{u, v\} \notin E$. A vertex cover B of a graph $G = (V, E)$ is a subset B of V such that $\{u, v\} \in E \Rightarrow u \in B$ or $v \in B$. Maximum and maximal independent sets are similarly defined as for clique. Minimum and minimal vertex covers are also similarly defined. The maximum clique size of a

graph G is called the *clique number* of G, usually denoted $\omega(G)$. A proper coloring of a graph $G = (V, E)$ is a function $f : V \mapsto \mathbb{N}$ such that $\{u, v\} \in E \Rightarrow f(u) \neq f(v)$. For each $v \in V$, $f(v)$ is called the *color* of v under f. The *chromatic number* of a graph G is the minimum number of colors over all proper colorings of G, usually denoted $\chi(G)$. The complement $\overline{G}$ of graph $G = (V, E)$ is the graph $\overline{G} = (V, \overline{E})$ where $\overline{E} = \{\{u, v\} | u \in V, v \in V, \{u, v\} \notin E\}$.

It is not hard to see that a maximum clique of a graph G is a maximum independent set of its complement graph $\overline{G}$. The complement of a maximum clique of a graph G is a minimum vertex cover of its complement graph $\overline{G}$. The chromatic number of a graph G is an upper bound of its clique number since every proper coloring of a clique has to assign a unique color to each member of the clique.

In the computational complexity world, $\mathcal{NP}$ is the set of languages that can be decided by a non-deterministic Turing machine in polynomial time. Equivalently, $\mathcal{NP}$ is the set of languages that can be verified by a polynomial time algorithm. A language L is called $\mathcal{NP}$-hard if every language in $\mathcal{NP}$ can be reduced to L in polynomial time. A language L is called $\mathcal{NP}$-complete if it is both $\mathcal{NP}$-hard and in $\mathcal{NP}$. The maximum clique problem, maximum independent set problem, minimum vertex cover problem and minimum coloring problem are all $\mathcal{NP}$-hard graph optimization problems. However, in the fixed parameter tractability world, they are not computationally equivalent. When considering the decision versions of these $\mathcal{NP}$-hard problems, we consider each input consists of a graph G and a parameter k. These are all natural parameterized problems. A parameterized problem (L, k) is said to be *fixed parameter tractable* (FPT) if it can be decided in time $f(k)|x|^c$ for every input (x, k) where f is a function that only depends on k and c is a constant. In this sense, the vertex cover problem is shown to be in FPT while the clique and independent set problems are believed not to be in FPT. Actually, they are shown to be W[1]-hard in the parameterized complexity world. The coloring problem remains $\mathcal{NP}$-hard for $k > 2$ thus is not in FPT unless $\mathcal{P} = \mathcal{NP}$.

1.2 Differential Analysis of Case/Control Microarray Gene Expression Data

As we have noted above, case/control data serve as important sources for bioinformatics research. A differential metric, by which we mean a difference between two values calculated on two groups of data (usually two vectors), can often be effective in identifying genes of interest from case/control data. We briefly review a traditional differential metric and introduce two novel differential metrics for case/control data analysis as our contributions with details in Chapter 2.

1.2.1 Differential Expression

The analysis of microarray gene expression data often involves identifying those genes whose expression levels change significantly between two sample groups. For example, we might want to ask which genes are up-regulated or down-regulated between treatment and control groups in order to understand the effects of a drug. The design of experiments is often such that each sample group contains several replicates. The group expression level for a gene can be summarized as the mean of the expression levels across the replicates of the group. Thus, differential expression analysis is a comparison of means. Some form of t test is usually performed for this situation.

1.2.2 Two Alternative Differential Metrics based on Variability to Identify Disease Related Genes

Although differential expression is a very effective metric, it nonetheless could miss certain important genes. We will investigate two alternative differential metrics based on entropy and coefficient of variation to identify disease related genes. These are covered in Chapter 2. The concept of entropy has played important roles in various fields of sciences, such as physics and information theory. In classical thermodynamics, the theory of entropy change is the foundation of thermal machines

such as heat engines and refrigerators. In information theory and cryptographic systems, entropy serves as a useful analytical tool for the efficiency and security of the systems. Coefficient of variation is a standard statistical measure of variability and has been used in various application domains. We will demonstrate the effectiveness of these two metrics by experiments on microarray gene expression data from 16 diseases' case-control datasets. We will also show how we developed an R package to calculate the differential values and p-values of the differential metrics.

1.3 Gene Co-Expression Networks and Clustering Analysis

The combinatorial object *graph* defined above is a very effective tool to model the relationships among a set of real world entities. When a graph is used to represent such relationships, it is also often called a *network*. In this section we review the various types of networks that can be derived from biological gene expression data and the clustering analysis commonly performed on those networks.

1.3.1 Types of Networks

Naswa [72] gave a good summary of various types of networks that can be extracted out of gene expression data obtained from microarrays. These include Boolean networks, Bayesian networks and relevance networks.

Liang et al [65] developed a mutual information based algorithm to extract gene regulatory networks from gene expression data, assuming that genes are present in two discrete states (on and off). The disadvantage of this method is that the assumption of binary state itself is biologically unrealistic and makes Boolean networks fall short of intermediary stages of gene expression.

Bayesian networks use directed acyclic graphs (DAG) to establish probabilistic relations among the genes. The ability to estimate the most probable directional

relationships is a major advantage of Bayesian networks [35]. However, heuristic algorithms or prior knowledge of networks are often employed to build Bayesian networks because learning Bayesian networks are computationally prohibitive [22].

Relevance networks are built by first employing similarity measures such as correlation and mutual information among all possible gene pairs in a transcription dataset to obtain a weighted network and then filtering it by employing an appropriate threshold to retain only those associations above the threshold [15, 60, 99]. Graph algorithms can then be utilized to extract dense or highly connected subgraphs from these relevance networks.

1.3.2 Clustering Methods

Clustering is used to aggregate observed entites into groups (called *clusters*) based on measures for evaluating similarity between observations. In a biological context, clustering is commonly used to group genes and/or samples for analysis of gene expression data, with the entities having similar expression profiles being grouped into the same or closer clusters. The measures used to determine the distance between observations and clusters include Euclidean distance, Manhattan distance, correlation, Kullback-Leibler divergence, etc [54]. Correlations measure the linear or non-linear similarity between expression patterns of genes or samples. The most frequently used measure is Pearson correlation. Another correlation measure is the non-parametric Spearman's rank correlation. It uses the relative ranks of the observations to perform calculations instead of the actual values.

Clustering methods can be categorized into *supervised* and *unsupervised* clustering [25]. Supervised clustering methods require prior knowledge of number of clusters in the data. They then allocate the genes into one of the clusters. The number of clusters is derived from the data itself in unsupervised clustering [3].

In clustering microarray gene expression data, the K-means clustering [91], hierarchical clustering [29] and fuzzy clustering [12, 39] are conventional. K-means

clustering groups the observations into a user specified number (K) of clusters based on the distance of each observation to the means of the clusters.

In hierarchical clustering all observations are grouped in a tree like structure without having to specify the number of clusters beforehand. The clusters can then be created by choosing a suitable level to cut the tree [29, 82]. Hierarchical clustering can go through the top down process or bottom up process, based on whether the observations are placed in one cluster or each observation is in its individual cluster initially. Hierarchical clustering has been used to cluster genes under different experimental conditions [85, 61].

Most clustering methods cluster observations into non-overlapping groups. In contrast, fuzzy clustering allows an observation to be associated with more than one cluster. This may be a more accurate method if a gene performs multiple functions in different groups.

1.3.3 Paraclique-based Clustering Methods

Jay et al [51] gave a systematic comparison of clustering algorithms. Graph-based clustering methods include WGCNA [105], CAST [9], CLICK [83], k-clique communities [75], NNN [47], maximal clique [106, 14, 1], and paraclique [18]. The first three of these are all heuristic approaches and the remaining depend on finding cliques. Paraclique was first introduced in [18] as a technique to construct dense subgraphs to overcome the weakness of cliques in dealing with noises in the data. A *paraclique* is a clique plus some additional vertices that are highly connected to the clique. The starting clique is usually a maximum clique of the input graph. The added vertices are those omitted out of the clique due to noise. The paraclique introduced in [18] is constructed using two level loops. It was simplified in [27] as the following algorithm, which has only an outer loop.

```
compute_paraclique(graph G = (V,E), glom factor g,vertex set C)
P = C
```

```
for each v in V-P {
if v is adjacent to at least g members of C { P = P U {v} }
}
return P
```

Figure 1.1 Paraclique Algorithm

With some type of iterative process, the function *compute_paraclique* can be used to find multiple dense subgraphs. The approach that has been used in [27] is to compute a maximum clique C of G, extend it to P with the function *compute_paraclique* in Figure 1.1. Then remove P from G and repeat until the graph G becomes empty or its maximum clique size falls below some threshold (a user supplied parameter). The glom factor there specifies an absolute number of vertices in C to which a glommed vertex must be adjacent, and since it does not scale with the size of C, it is a poor parameter. Setting the glom factor as $g = |C| - a$, where $|C|$ is the maximum clique size of the remaining graph and a is a small positive integer, would provide a better scaling alternative.

The paraclique algorithm in Figure 1.1 requires the user to supply a glom factor, which could pose the following problems. First, setting the glom factor to $|C| - a$ does not scale perfectly because it effectively specifies the missing number of edges allowed. Second, the paraclique algorithm does not distinguish between those vertices that can be added but have different relative distances to the set C. Intuitively vertices with more connections to the clique should be favored.

```
compute_phased_paraclique(graph G = (V,E), ratio R, vertex set C)
P = C
a = 1
while (a <= R*|P|) {
P = compute_paraclique(G, |P|-a, P)
a = a + 1
}
```

```
return P
```

Figure 1.2 Phased Paraclique Algorithm

An iterative algorithm *compute_phased_paraclique* is shown in Figure 1.2. It allows the user to set some reasonable stopping criteria for paraclique growth by the ratio parameter R, resulting in better control over augmentation. It can be seen that added vertices must be adjacent not only to sufficient numbers of vertices of the original maximum clique but also to vertices added to the paraclique at prior iterations. While giving priority to closer vertices, it can also bring in vertices that are still close enough to the original clique in later iterations. Phased paraclique halts expansion based on the parameter R, which is an upper bound of the percentage of the allowed missing edges in P for the newly added vertex. [27] also compares the paraclique algorithm in Figure 1.1 and the phased paraclique algorithm in Figure 1.2 by experiments with a transcriptomic graph, showing the number of resulting paraclique, their sizes and densities.

In [51] paraclique has been shown superior to traditional clustering methods for biological data analysis. We provide the density analysis for paracliques in Chapter 3. This serves as a performance guarantee of paracliques in extracting dense cores from graphs.

1.3.4 Evaluation of Gene Cluster Qualities

Gene ontology (GO) is a database consisting of information that defines the functions of genes. The online DAVID Bioinformatics Resources [45, 44] provides GO enrichment analysis of gene clusters.

Other than providing systematic description of gene functions, GO also automates the process of functional analysis of groups of genes, which provides the basis for studies of gene enrichment and one of the evaluation criteria for gene clustering methods.

Typically GO database is employed to identify GO-enriched categories in a set of genes by some analytical method, such as differential analysis and clustering. Statistical methods such as hypergeometric test, Fishers exact test and Chi-square test are employed to detect the GO categories [11, 67, 8] that are overrepresented in the gene set being queried.

In Chapter 2 we use GO enrichment extensively to study the effectiveness of various differential metrics.

1.4 An Empirical Study of Graph Optimization through Transformational Approaches

$\mathcal{NP}$-complete problems are important both theoretically and practically. Huge amounts of time and efforts have been devoted to the development of solvers for them, both exactly and approximately. Since these problems are mutually reducible to each other, algorithms for one of them can be used to solve all the others from a theoretical sense, when an explicit reduction is given. In this dissertation we will only consider exact algorithms for $\mathcal{NP}$-hard graph optimization problems. Based on any general polynomial time reduction between two decision problems or between two optimization problems, the exact solution for the reduced problem can be used to recover the exact solution for the original problem, which constitutes the basis for solving a problem indirectly by using solvers for other problems. Satisfiability (SAT) and Integer Linear Programming (ILP) problems are two hub problems in the $\mathcal{NP}$-complete world and many advanced solvers have been developed for them over the years. It would be very interesting to investigate whether they can be used to solve other $\mathcal{NP}$-complete problems more efficiently than solving those problems using direct algorithms. It would be even more instructive to identify the instance features corresponding to particular regions of input space where such transformations are effective alternatives to direct optimization. We have chosen the well-known graph

optimization problem MaxCLIQUE (the maximum clique problem) to conduct an empirical study of the transformational approaches based on SAT and ILP. The results are covered in Chapter 4.

Chapter 2

Differential Shannon entropy and differential coefficient of variation: alternatives and augmentations to differential expression in the search for disease-related genes

(Note: The contents of this chapter are largely from the published paper [100].)

DNA microarray has become an important technology to measure the expression levels of a large number of genes simultaneously. Since the advent of microarray technology, differential expression has become a standard tool for analyzing case-control transcriptomic data. It has proved invaluable in characterizing the molecular mechanisms of disease by distinguishing relevant genes. Nonetheless, the expression profile of a gene across different experimental conditions can be perturbed in ways that leave the expression level unvaried, while a biological effect is nevertheless present. We investigate and analyze differential Shannon entropy and differential coefficient of variation, two alternative techniques for identifying genes of interest.

Ontological analysis across sixteen human disease datasets demonstrates that these alternatives are effective at identifying disease-related genes not found by mere differential expression alone. Because the two alternate techniques are based on somewhat different mathematical formulations than differential expression, they tend to produce somewhat different gene lists. Moreover, each may pinpoint genes completely overlooked by the other. Thus, these two measures based on entropy and coefficient of variation respectively can be used to replace or better yet augment standard differential expression computations.

2.1 Introduction

One of the key goals of high-throughput transcriptomic experiments and associate data analysis is to identify genes relevant to disease, treatment or other factors. Such a gene is oftentimes viewed as one whose expression is altered in the presence of some relevant stimulus. Hence, mRNA microarray experiments are frequently designed with a case (treatment, stimulus) population and a control (healthy, unaffected) population. Differential expression has long been the standard technique to identify up-regulated or down-regulated genes in such experiments [46, 63, 87]. It continues to be applied as new technologies such as RNA-Seq emerges on the horizon [4, 73]. Although differential expression has proven to be a tremendously valuable aid, it is limited in that it only detects altered *levels* of expression. Changes to the expression profile between case and control with little or no corresponding alteration in mean expression level will often remain unspotted.

We discuss and analyze two variability-based techniques to identify genes of interest in case-control data. One of them is based on the notion of Shannon entropy. The other harnesses the coefficient of variation. Both entropy and variation focus on expression variability on a normalized scale, while differential expression is based instead on simple mean differences. Over the analysis of a collection of various

biological datasets, we demonstrate that entropy and variation are able to identify genes relevant to disease but overlooked by differential expression.

Shannon entropy and coefficient of variation are widely used in a variety of application domains, from theoretical physics to computational chemistry and materials science. They have been applied in bioinformatics as well, most notably in statistical genetics and molecular biology. Entropy is derived from information theory [81]. It has been used, for example, to track gradual expression changes in cancer growth [10], as a measure of genetic diversity at levels ranging from gene expression to landscapes [84], as an estimate of the structural diversity of ecological species classifiers [69], as a measure of the robustness of gene regulatory networks [17], to accelerate feature elimination when classifying microarray expression data [36], and as a pre-processing filter on microarray expression data [56]. Coefficient of variation is a standard statistical measure. It has, for example, been used to assess variability of quantitative assays [79], as a predictor of risk sensitivity in animals [101], to analyse synaptic plasticity [31], and to compare diversity among workforces [7]. To the best of our knowledge, neither technique has been applied to analyze transcriptomic data in the differential context in which we shall employ them here.

2.2 Experimental data sets and procedures

We tested two differential metrics, one based on normalized Shannon entropy (SE) and the other on coefficient of variation (CV). We used publicly available mRNA gene expression data, and compared the results obtained by these two metrics to those obtained with differential expression(DE). Both SE and CV measure the variability or dispersion of data in a normalized fashion. They are defined as follows. Let $x_1, x_2, \ldots, x_n$ denote a list of n numbers and let $x = \sum_{i=1}^{n} x_i$ signify their sum. Then

the normalized Shannon entropy of this list is defined as:

$$SE = \frac{-\sum_{i=1}^{n} \frac{x_i}{x} \log_2 \frac{x_i}{x}}{\log_2 n} \tag{2.1}$$

and the coefficient of variation is defined as:

$$CV = \frac{s}{|\overline{x}|} \tag{2.2}$$

where $\overline{x} = x/n$ is the sample mean and $s = \sqrt{\sum_{i=1}^{n} (x_i - \overline{x})^2/(n-1)}$ is the sample standard deviation. In the context of gene expression data, this list of numbers represents the expression levels of a particular probe set over different biological samples.

Table 2.1: GEO Series number for the 15 public disease datasets

Dataset	GEO Series number
Asthma	GSE4302
Breast cancer	GSE10810
Chronic lymphocytic leukaemia	GSE8835
Colorectal cancer	GSE9348
Crohns disease	GSE6731
Lung adenocarcinoma	GSE7670
Multiple sclerosis	GSE21942
Obesity	GSE12050
Pancreatic tumour	GSE16515
Parkinsons disease	GSE20141
Prostate cancer	GSE6919
Psoriasis	GSE13355
Schizophrenia	GSE17612
Type 2 diabetes	GSE20966
Ulcerative colitis	GSE6731

We calculate SE and CV of each probe set separately for case and control. The case-control differences produce values we term differential Shannon entropy (DSE) and differential coefficient of variation (DCV). In the past the phrase "differential

entropy" has been used in information theory to mean a continuous extension from the discrete version [19], in a sense similar to differential calculus. That is not how we intend to employ the term here, of course. There are many ways to calculate DE. For example, one can use a t-test with Bonferroni correction. Methodological differences tend to be minimal, however, and henceforth for our purposes we simply took the magnitude of the difference between the mean expression level for case and the mean expression level for control. We applied our three metrics (DE, DSE and DCV) to 15 publicly available disease datasets obtained from the Gene Expression Omnibus (GEO) [28], and to one in-house allergic rhinitis dataset. We have chosen case-control data in a form easy to analyze, striving for a reasonable variety of complex human diseases. All data are from mRNA microarray experiments. The GEO series identifiers are shown in Table 2.2. The number of probe sets ranges from roughly 12,000 to 61,000. The number of samples in case and control ranges from 4 to 63.

For the 100 probe sets with the highest differential values for each of the three techniques, we performed Gene Ontology (GO) enrichment analysis with the online hypergeometric test provided by DAVID Bioinformatics Resources 6.7 [45, 44]. We used the lowest p-value among all enriched categories as an estimate of the relative efficacy of each technique. The choice of 100 as a cut-off is kind of arbitrary. But for the purpose of demonstrating efficacy of the new techniques, any reasonable cut-off that allows analysis of probe sets in the top range of each metric suffices.

2.3 Experimental results and discussion

Our results are centred on answering two questions. First, to what extent are DE, DSE and DCV surrogates for each other? In other words, do they tend to identify the same genes? Second, what is the relative effectiveness of each technique? Does each metric independently select genes relevant to the disease under study?

Figure 2.1 addresses the first question. Venn diagrams for each of the sixteen diseases depict the overlap between the top 100 genes chosen by each technique. Of

Table 2.2: The most enriched categories for each metric on each disease. For each of the 16 diseases, the most highly enriched GO category by the hypergeometric test for each of the three metrics is shown, along with its enrichment p-value

Dataset	DE Category & p-value	DSE Category & p-value	DCV Category & p-value
Allergic rhinitis	Signal p=8.5E-16	Signal p=6.0E-13	Signal p=6.2E-13
Asthma	Signal peptide p=1.2E-5	Secreted p=3.4E-11	Secreted p=2.0E-9
Breast cancer	Response to endogenous stimulus p=7.8E-13	Secreted p=1.5E-6	Signal p=2.2E-6
Chronic lymphocytic leukaemia	Immunoglobulin C1-set p=4.4E-21	Disulphide bond p=2.4E-15	Disulphide bond p=1.4E-14
Colorectal cancer	Acetylation p=9.8E-7	Signal p=2.6E-6	Signal p=5.1E-8
Crohns disease	SM00407:IGc1 p=7.6E-12	Defence response p=7E-8	Micro-some p=3.9E-5
Lung adenocarcinoma	Immunoglobulin; major histocompatibility complex, conserved site p=6.2E-10	Secreted p=7.9E-12	Signal p=8.3E-11
Multiple sclerosis	Translational elongation p=5.2E-26	Immune system process p=8.9E-6	Erythrocyte differentiation p=4.1E-4
Obesity	Antimicrobial p=9E-7	Defence response p=4.3E-9	Oxygen transport p=6.5E-8
Pancreatic tumour	Digestion p=3.1E-16	Digestion p=4.9E-9	Response to chemical stimulus p=2.7E-6
Parkinsons disease	Membrane associated complex p=1.9E-6	Organic acid: sodium symporter activity p=3.4E-3	Alternative splicing p=9.3E-4
Prostate cancer	Translational elongation p=8.6E-38	Muscle protein p=1.9E-18	Contractile ber part p=7E-17
Psoriasis	Secreted p=1.7E-15	Secreted p=4.1E-10	Secreted p=4.9E-11
Schizophrenia	Acetylated amino end p=3.1E-11	Secreted p=2.1E-7	Extracellular region p=1.4E-7
Type 2 diabetes	Pancreas p=3.7E-20	Signal p=2.3E-8	Signal p=1.1E-4
Ulcerative colitis	SM00407:IGc1 p=8.3E-14	Signal p=3.7E-15	Inflammatory response p=2.4E-16

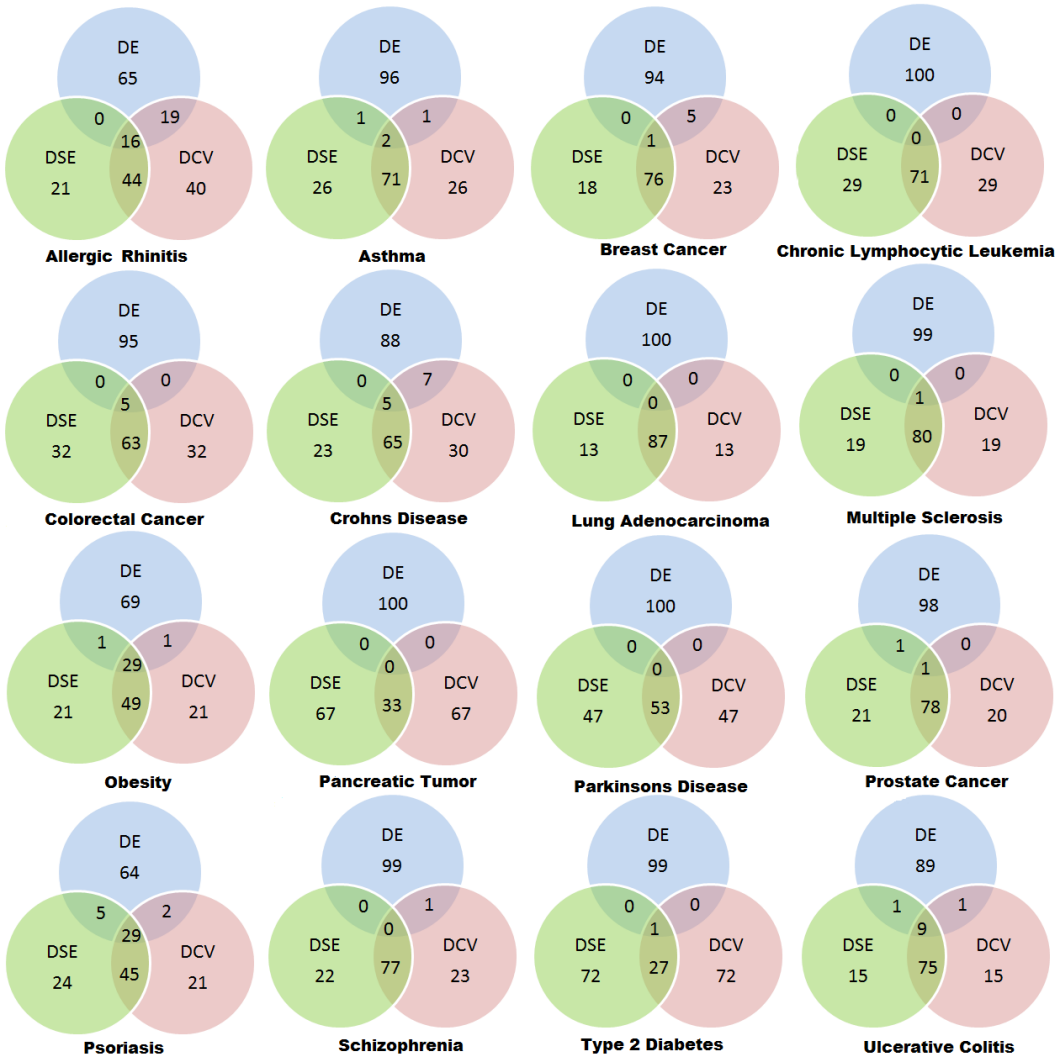

Figure 2.1: Commonality of genes over three metrics. In the Venn diagrams, one for each disease, each circle represents the top 100 genes selected by one of the three metrics under study. In most cases, few of the top DE genes are also ranked near the top for DSE or DCV. On the other hand, DSE and DCV have a majority of top-ranked genes in common for all but two diseases.

the genes found by the two new metrics, at most 36 of 100 would have been found by differential expression (in the Psoriasis dataset). Most of the diseases exhibit far fewer genes in common, 11 of 16 diseases having six or fewer genes shared between DSE-DCV and DE. Moreover, it is not the case that genes found by DSE and DCV are highly ranked but just barely missed by DE. See Table 2.3. DE thus seems to be an extremely weak predictor for genes found by DSE and DCV. As one might expect, however, the two variation-based approaches tend to produce more similar results. DSE and DCV show strong overlap. More than half their top 100 genes are shared in all but two diseases (type 2 diabetes and pancreatic tumour); more than two-thirds are shared in 12 of 16 diseases.

Table 2.3: The DE ranking of genes ranked in the top 100 by DSE and DCV, but not ranked in the top 100 by DE. From this we conclude that DE would have missed great numbers of disease-related genes identified by DSE and DCV

Dataset	Gene count	Minimum position	Maximum position	Average position	Standard deviation
Allergic rhinitis	48,803	102	37,051	3037	5294
Asthma	54,675	116	43,008	8907	7728
Breast cancer	18,382	151	18,144	5340	4749
CLL	22,283	229	22,110	7954	6494
Colorectal cancer	54,675	109	53,921	20,255	14,808
Crohns disease	12,625	144	12,283	4262	3400
Lung adenocarcinoma	22,283	236	17,572	5636	4049
Multiple sclerosis	54,675	141	23,661	7153	6399
Obesity	44,290	888	38,157	8064	9161
Pancreatic tumour	54,613	416	27,179	8377	5937
Parkinsons disease	54,675	1213	53,411	12,611	11,622
Prostate cancer	12,625	129	12,613	4567	3733
Psoriasis	54,675	102	49,861	5303	8532
Schizophrenia	54,675	308	54,518	21,731	17,796
Type 2 diabetes	61,294	328	61,274	30,388	18,305
Ulcerative colitis	12,625	122	12,257	3393	2952

We address the second question by examining first the GO enrichments of the top 100 genes produced by each method, then the enrichment p-value by rank for each method. Figure 2.2 shows the lowest p-value among all GO categories for each

disease over each of the three metrics. DE tends to have the most significant p-values, although in five cases it is surpassed by DSE and/or DCV (asthma, colorectal cancer, lung adenocarcinoma, obesity, and ulcerative colitis). Figure 2.3 plots the p-value of the most enriched GO category by gene rank for the allergic rhinitis dataset. We selected sets of 100 genes at different rank positions for each method and performed a GO enrichment test for each set. In all three methods, the top sets of 100 genes had markedly lower p-value than all other sets of 100 genes. And all three methods show notably decreased functional enrichment after the first few hundred genes. It therefore appears that each metric is independently finding disease-relevant genes.

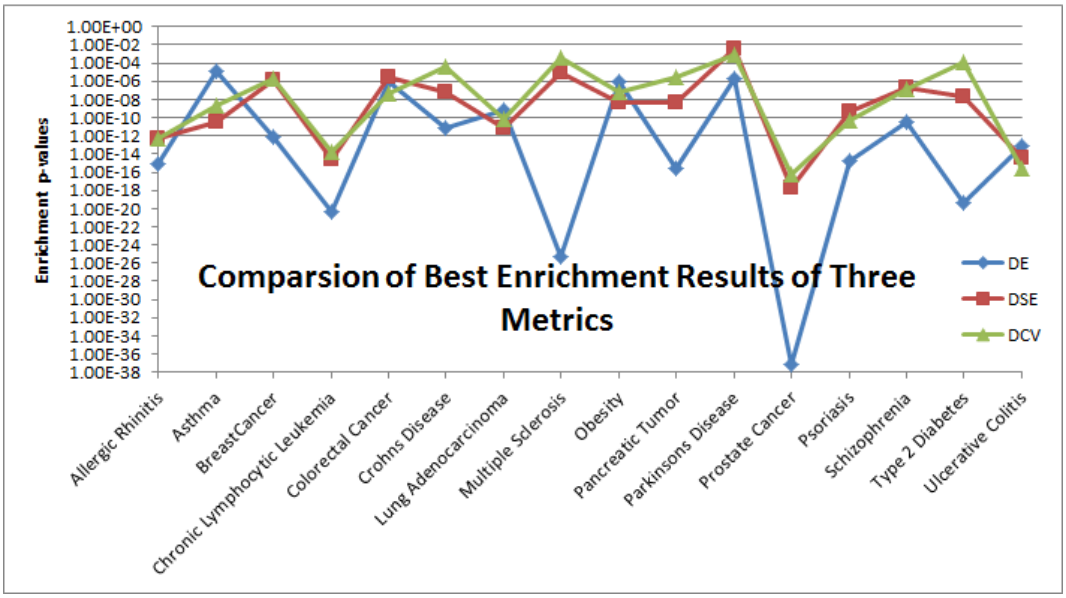

Figure 2.2: Comparison of gene enrichment over three metrics. The p-value of the most enriched GO category for each disease is shown for DE, DSE and DCV.

We next examined the enriched GO categories for the three different methods to check for relevance to specific diseases. We highlight three examples to illustrate the effectiveness of the variation-based techniques. In the asthma dataset (GSE4302), the most enriched GO category is secreted for both DSE (p = 3.37E-11) and DCV (p = 2.02E-09). Yet the category is not enriched at all in the top 100 DE genes. In the Crohns disease dataset (GSE6731), the category response to wounding is the second most enriched for both DSE (p = 3.43E-7) and DCV (p = 4.15E-5), but is

not enriched in the top 100 DE genes. And in the lung adenocarcinoma dataset (GSE7670), the category extracellular region is highly enriched in the top 100 genes for all three metrics, DE ($p = 2.28\text{E-}8$), DSE ($p = 2.81\text{E-}11$) and DCV ($p = 9.36\text{E-}11$). Yet the latter two methods have no overlap at all with DE for this disease (see Figure 2.1).

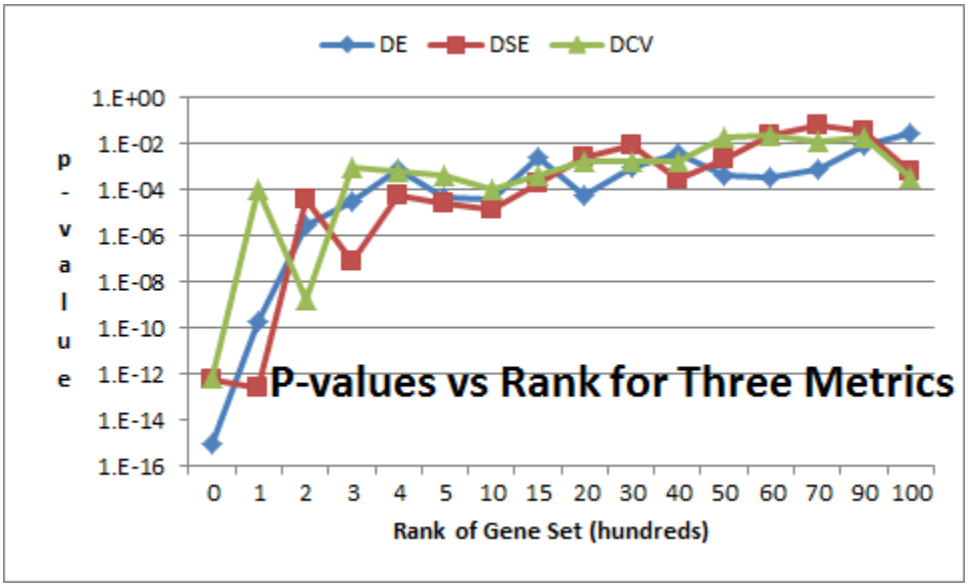

Figure 2.3: Enrichment by rank over three metrics. For the allergic rhinitis dataset, the lowest GO enrichment p-value is shown for sets of 100 genes at various points in the rankings. The top 100 genes are shown at point 0, genes ranked 101∼200 at point 1, genes ranked 2001∼2100 at point 20 and so on. All three metrics show a sharp drop-off in functional enrichment after the top few hundred genes,suggesting that each metric is indeed identifying relevant genes. Other diseases exhibit similar results.

2.4 Theoretical Analysis of the Two Alternative Metrics

In this section we prove the possible range of entropy and a variation of coefficient of variation of an arbitrary list of positive numbers.

2.4.1 Bounds of Entropy

Theorem 2.1. *Let $x_1, x_2, \ldots, x_n$ denote a list of positive numbers and let $x = \sum_{i=1}^{n} x_i$ signify their sum. Then the normalized Shannon entropy of this list defined as $SE = \frac{-\sum_{i=1}^{n} \frac{x_i}{x} \log_2 \frac{x_i}{x}}{\log_2 n}$ satisfies $0 \leq SE \leq 1$.*

Proof. We have $f(x) = \log_2 x$ is a strictly concave funtion on the interval $(0, \infty)$. Suppose $a_i > 0, 1 \leq i \leq n$, and $\sum_{i=1}^{n} a_i = 1$. According to Jensen's inequality ([88]), we have

$$\sum_{i=1}^{n} a_i f(y_i) \leq f(\sum_{i=1}^{n} a_i y_i)$$

where $y_i \in (0, \infty), 1 \leq i \leq n$. Now let $a_i = x_i/x, y_i = 1/a_i$, we have

$$\begin{aligned} SE &= \frac{\sum_{i=1}^{n} a_i f(y_i)}{\log_2 n} \\ &\leq \frac{f(\sum_{i=1}^{n} a_i y_i)}{\log_2 n} \\ &= \frac{\log_2 n}{\log_2 n} = 1 \end{aligned}$$

Clearly $SE \geq 0$, and the theorem is proved. □

From the proof we can see that $SE = 1$ if and only if $x_1 = \ldots = x_n$. If we allow $x_1, \ldots, x_n$ to be non-negative and further stipulate that $0 \log_2 0 = 0$, then $SE = 0$ if and only if exactly one of $x_1, \ldots, x_n$ is positive and the rest of them are all 0.

2.4.2 Bounds of Variation

From the definition of coefficient of variation, it is clear that $CV \geq 0$ and $CV = 0$ if and only if $x_1 = \ldots = x_n$. If we fix the mean of the list $x_1, \ldots, x_n$ and increase the standard deviation by moving the numbers further from each other, we can see that CV has no upper bound. However, if we define the normalized coefficient of variation as $NCV = \frac{CV}{\sqrt{n}}$, then this quantity is bounded. In fact, we have the following:

Theorem 2.2. *Let $x_1, x_2, \ldots, x_n$ denote a list of positive numbers and let CV signify their coefficient of variation. Then the normalized coefficient of variation defined as $NCV = \frac{CV}{\sqrt{n}}$ satisfies $0 \leq NCV \leq 1$.*

Proof. We have

$$\begin{aligned}
CV &= \frac{\sqrt{\sum_{i=1}^{n} (x_i - \overline{x})^2/(n-1)}}{\overline{x}} \\
&= \frac{\sqrt{\sum_{i=1}^{n} (x_i^2 + \overline{x}^2 - 2x_i\overline{x})}}{\sqrt{n-1}\overline{x}} \\
&= \frac{\sqrt{\sum_{i=1}^{n} x_i^2 + n\overline{x}^2 - 2n\overline{x}^2}}{\sqrt{n-1}\overline{x}} \\
&= \frac{\sqrt{\sum_{i=1}^{n} x_i^2 - n\overline{x}^2}}{\sqrt{n-1}\overline{x}} \\
&= \frac{\sqrt{\sum_{i=1}^{n} (x_i/\overline{x})^2 - n}}{\sqrt{n-1}} \\
&= \frac{\sqrt{n^2 \frac{\sum_{i=1}^{n} x_i^2}{(\sum_{i=1}^{n} x_i)^2} - n}}{\sqrt{n-1}} \\
&\leq \frac{\sqrt{n^2 - n}}{\sqrt{n-1}} \\
&= \sqrt{n}
\end{aligned}$$

thus $NCV = \frac{CV}{\sqrt{n}} \leq 1$ and the theorem is proved. □

From the proof we can see that if we allow $x_1, \ldots, x_n$ to be non-negative then $NCV = 1$ if and only if $SE = 0$ and $NCV = 0$ if and only if $SE = 1$. From this we can see that SE and CV tend to vary in opposite directions.

2.5 Experimental Details

In deriving the results of our published paper [100], we calculated the three differential metrics by Excel. We imported the properly formated raw gene expression data files into Excel and programmed the differential metrics using built-in Excel functions. During the experiments we found that Excel can be quite slow in loading large data files. We realized that it may be more convenient for ourselves and other scientists to have an easy-to-use tool to calculate these differential metrics. That is our topic for the next section.

2.6 EntropyExplorer: An R package for computing and comparing differential Shannon entropy, differential coefficient of variation and differential expression

(Note: This section is largely taken from a draft paper under review.) Differential Shannon entropy (DSE) and differential coefficient of variation (DCV) are effective metrics for the study of gene expression data. They can serve to augment differential expression (DE), and can be applied in numerous settings whenever one seeks to measure differences in variability rather than mere differences in magnitude. A general purpose, easily accessible tool for DSE and DCV would help make these two metrics available to data scientists. Automated p-value computations would additionally be useful, and are often easier to interpret than raw test statistic values alone.

EntropyExplorer is an R package for calculating DSE, DCV and DE. It also computes corresponding p-values for each metric. All features are available through a single R function call. Based on extensive investigations in the literature, the Fligner-Killeen test was chosen to compute DCV p-values. No standard method was found to be appropriate for DSE, and so permutation testing is used to calculate DSE p-values.

EntropyExplorer provides a convenient resource for calculating DSE, DCV, DE and associated p-values. The package, along with its source code and reference manual, are freely available from the CRAN public repository at http://cran.r-project.org/web/packages/EntropyExplorer/index.html

2.6.1 Background

Normalized Shannon entropy (SE) and coefficient of variation (CV) are used to measure the variability or dispersion of numerical data. Such variability has potential utility in numerous application domains, perhaps most notably in the analysis of high throughput biological data. Variability has been applied, for example, to study gene expression data in the context of human disease [43]. Increased entropy in particular, in both gene expression and protein interaction data, has been observed to be a characteristic of cancer [102]. Numerous other examples typify the utility of entropy [10, 84, 69, 17, 36, 56] and coefficient of variation [79, 101, 31, 7].

Shannon entropy is famously rooted in information theory [81]. To avoid confusion, we emphasize that we use the term differential Shannon entropy to denote a difference between two Shannon entropy values. This is distinct from information-theoretic terminology, in which differential entropy often means the entropy of a continuous, rather than a discrete, random variable [19].

We are particularly interested in differential analysis. In [100], we studied differential Shannon entropy (DSE) and differential coefficient of variation (DCV), and found them highly effective in identifying genes of potential interest not found by differential expression (DE) alone. DSE and DCV are also applicable to other types of biological data as well, such as that produced by RNA-seq technologies, although the usual caveats about careful interpretation apply. The usefulness of DSE and DCV is of course not limited to biological data. They may be applied to any numerical data for which normalized measures of differential variability are relevant.

2.6.2 Implementation

EntropyExplorer is implemented in R [78]. All features are wrapped into a single function call, which takes as input up to eight arguments. Two of these arguments are numerical matrices, with identical labels for each row. The output is a matrix with two, three or five columns that contains in each row two SE, CV or mean values and a DSE, DCV or DE value and/or two p-values, one raw and one adjusted. Output rows can be sorted by value, raw p-value or adjusted p-value, and can be filtered to show only the top-ranked rows.

Permutation testing for DSE is accomplished with the help of the R function *sample.int*. The default number of tests to be employed is set to 1000, which the user can override. The p-value for DCV is calculated by applying the Fligner-Killeen test for homogeneity of variances, implemented via the R function *fligner.test*, to the log-transformation of the input data. The R function *t.test* is used to find a p-value for DE. Adjusted p-values are calculated using the *p.adjust* function in R, which provides false discovery rate and multiple testing corrections. A more thorough explanation of p-value calculations is provided later in the discussion section.

EntropyExplorer checks that all matrix entries are positive. This is because calculations of a DSE value/p-value and a DCV p-value involve taking logarithms, which are undefined on data containing zeros or negative values. Also, CV becomes less meaningful when means approach zero or are negative. Experimental data may be noisy, however, and so EntropyExplorer provides mechanisms to handle non-positive values. An optional two-value argument permits the user to add a positive bias to all elements of one or both matrices prior to performing any other calculations. The argument can also be set to make this adjustment automatically, based on the least non-positive value in each matrix.

We refer to equations 2.1 and 2.2 for definitions of SE and CV. As noted in Chapter 2, normalized Shannon entropy falls in the range [0, 1]; DSE therefore also falls in

the range [0, 1]. Lower (higher) SE corresponds to more (less) variability. CV falls in the range $[0, \infty)$; DCV therefore also has a range of $[0, \infty)$.

2.6.3 Application

EntropyExplorer is invoked as follows:

```
EntropyExplorer(expm1, expm2, dmetric, otype,
                          ntop, nperm, shift, padjustmethod)
```

We refer the reader to the package documentation, included in both the supplemental materials and the project webpage, for a detailed description of all arguments and options. Included with the package is a sample mRNA microarray dataset obtained from the Gene Expression Omnibus (GEO) [28]. This dataset, GSE10810, contains case/control data on breast cancer [76]. Figures 3.1 and 3.2 provide example uses of EntropyExplorer on this data.

Figure 3.1: The output of EntropyExplorer on breast cancer data. The numerical matrices m1 and m2 have been read into R. The function call has specified "dse" for differential Shannon entropy, "v" for value, and 10 to return the top 10 values.

```
> EntropyExplorer(m1,m2,dmetric="dse",otype="v",ntop=10)
            SE(expm1) SE(expm2) SE(expm1)-SE(expm2)
228245_s_at 0.4585632 0.9749417          -0.5163785
205220_at   0.4347483 0.9494563          -0.5147080
213711_at   0.4658234 0.9705819          -0.5047585
209242_at   0.4757616 0.9597454          -0.4839837
203908_at   0.4971761 0.9460891          -0.4489130
205030_at   0.4183439 0.8599655          -0.4416216
227282_at   0.5773323 0.9901547          -0.4128224
205067_at   0.5584833 0.9406934          -0.3822100
223623_at   0.6033164 0.9801690          -0.3768526
```

```
203824_at    0.5562821 0.9249965          -0.3687144
```

Figure 3.2: Another use of EntropyExplorer on breast cancer data. The function call has specified "dcv" for differential coefficient of variation, "bv" to specify both value and p-value, and to sort by value, and 12 to return the top 12 rows.

```
> EntropyExplorer(m1, m2, dmetric="dcv", otype="bv",ntop=12)
        CV(expm1) CV(expm2) CV(expm1)-CV(expm2)  p-value fdr p-value
205220_at    3.970528 0.5873201    3.383208 0.8222579620 0.876318753
213711_at    3.789072 0.5350799    3.253993 0.2336725875 0.354813275
228245_s_at 3.603529 0.4299952    3.173533 0.0118800876 0.046483561
209242_at    3.455212 0.5763780    2.878834 0.0003329751 0.005779234
205067_at    3.399742 0.6848428    2.714899 0.3342106218 0.459651180
207302_at    3.325542 0.8120794    2.513462 0.0314091253 0.087452672
227282_at    2.765104 0.2722402    2.492864 0.1106427072 0.208619781
203908_at    3.066962 0.5786523    2.488309 0.0541907559 0.126061057
226147_s_at 3.530525 1.0483199    2.482205 0.2123513195 0.330828202
223623_at    2.835876 0.3818841    2.453992 0.0001887257 0.004363719
230285_at    3.059179 0.6062872    2.452892 0.0797618163 0.165595404
205752_s_at 2.836653 0.4284138    2.408240 0.7468782641 0.821021185
```

2.6.4 Kolmogorov-Smirnov Test

In this section we briefly review Kolmogorov-Smirnov Test, which provides one of the logical foundations for the p-value calculation method for DSE.

In the field of statistics there is often the need to test a set of data against certain probability distribution. Suppose there is a random sample $X_1, X_2, \ldots, X_n$ from some unknown continuous distribution with c.d.f. $F(x)$. We wish to test the simple null hypothesis H_0 that $F(x)$ is actually a particular continuous c.d.f. $F^*(x)$ against the

general alternative H_1 that the actual c.d.f. $F(x)$ is not $F^*(x)$. The Kolmogorov-Smirnov test can be used to test such a hypothesis. First we review some concepts and theorems.

Sample Distribution Function [20]: Let $x_1, x_2, \ldots, x_n$ be the observed values of a random sample $X_1, X_2, \ldots, X_n$. For each real number x define the value $F_n(x)$ as the proportion of observed values in the sample that are less than or equal to x. In other words, if exactly k of the observed values in the sample are less than or equal to x, then $F_n(x) = k/n$. The function $F_n(x)$ defined in this way is called the sample distribution function.

Now let $F(x)$ denote the c.d.f. of the distribution from which the random sample $X_1, X_2, \ldots, X_n$ was drawn. According to the law of large numbers we have that as $n \to \infty$, the proportion $F_n(x)$ of observations in the sample that are less than or equal to x will converge in probability to $F(x)$, i.e. $F_n(x) \xrightarrow{p} F(x)$ for all real x.

Glivenko-Cantelli Lemma [20]: Let F_n be the sample c.d.f. from an i.i.d. sample $X_1, X_2, \ldots, X_n$ from the c.d.f. F. Define the random variable D_n as $D_n = \sup_{-\infty<x<\infty} |F_n(x) - F(x)|$. Then D_n converges in probability to 0, i.e. $D_n \xrightarrow{p} 0$.

Now the Kolmogorov-Smirnov test will use the statistic $D_n^* = \sup_{-\infty<x<\infty} |F_n(x) - F^*(x)|$ and based on a theorem of A. N. Kolmogorov and N. V. Smirnov, if the null hypothesis H_0 is true, then for each given value $t > 0$, we have

$$\lim_{n\to\infty} Pr(\sqrt{n}D_n^* \leq t) = H(t) = 1 - 2\sum_{i=1}^{\infty}(-1)^{i-1}e^{-2i^2t^2}$$

and for any specified level of significance α_0, we can choose the critical value c to be the $1-\alpha_0$ quantile $H^{-1}(1-\alpha_0)$ of the distribution H and we reject H_0 if $\sqrt{n}D_n^* \geq c$.

2.6.5 Discussion

In addition to calculating DSE, DCV and DE, EntropyExplorer can calculate both raw and adjusted p-values for each of these differential metrics. ANOVA-based tests

are the standard way to obtain differential expression p-values. We therefore use a t-test for this purpose. Certainly more sophisticated methods exist. See, for example, [66, 95]. For DCV p-values, we observe that 11 tests of equal relative variation were compared in [23], with the conclusion that the Fligner-Killeen test [33] is usually the most appropriate. It strikes a balance between type I and type II errors, and is robust to non-normal distributions.

Obtaining reliable p-values for DSE proved much more challenging. We found no known method in the literature specific to DSE p-values. We therefore investigated the extent to which SE is correlated to variance. A high correlation would suggest that they may be proxies for each other, in which case the p-value of an F-test or some derivation thereof might serve as suitable estimate of the DSE p-value. Unfortunately, correlations between SE and variance, or SE and a function of variance, were not high enough to justify using one as a surrogate for the other. Table 2.4 shows the correlation between SE and variance, and SE and a function $1/2\ln(2\pi eV)$ of variance as an attempt to linearize the relationship, using the 16 datasets from [100]. The only notably high correlation is found in the obesity dataset. The obesity data, however, contains a large number of missing values, rendering the high correlation less reliable. We conclude that standard statistical tests related to variance do not appear to be suitable for testing DSE.

We also examined the distribution of DSE on the 16 datasets, with the goal of empirically determining a suitable reference distribution for DSE. From this, we could then estimate p-values analytically. We applied the Kolmogorov-Smirnov (KS) test to compare the DSE distribution of each dataset to some of the more common reference distributions, such as normal, F, t, and chi-square. When performing a KS test, p-values can be overly sensitive to deviations from the reference distribution [40], so a D-statistic value below 0.1 was used to identify matching distributions. In our experiments, only the Parkinsons dataset produced a D-statistic below 0.1 when tested against a normal or standardized t distribution (Table 2.5). Figure 2.4 shows a sample distribution of DSE, in this case using the prostate cancer data.

Table 2.4: Correlations between SE and variance, and between SE and $1/2\ln(2\pi eV)$, on 16 microarray gene expression datasets.

Dataset	Correlation Between SE and Variance		Correlation Between SE and $1/2\ln(2\pi eV)$	
	case	control	case	control
Allergic Rhinitis	-0.5515	-0.5769	-0.9703	-0.9658
Asthma_GSE4302	-0.4272	-0.4677	-0.1924	-0.2004
BreastCancer_GSE10810	-0.3942	-0.3378	-0.1810	-0.1265
CLL_GSE8835	0.2251	0.2522	-0.08060	-0.06239
ColorectalCancer_GSE9348	0.3122	0.4454	-0.00863	0.02056
CrohnsDisease_GSE6731	-0.2826	-0.2380	-0.1664	-0.402
LungAdenocarcinoma_GSE7670	0.07254	0.3360	-0.0173	0.01049
MS_GDS3920	-0.3615	-0.332	-0.0515	-0.0559
Obesity_GSE12050	0.9998	0.999	0.1584	0.5420
Pancreas_GDS4102	-0.4137	-0.4455	-0.1331	-0.089
ParkinsonsDisease_GSE20141	-0.1732	-0.2554	-0.0024	-0.0155
ProstateCancer_GSE6919_GPL8300	0.2118	0.1552	-0.0562	-0.0699
Psoriasis_GSE13355	-0.6386	-0.6554	-0.52	-0.6779
Schizophrenia_GSE17612	0.3632	0.391	0.01695	0.02352
T2D_GSE20966	-0.6006	-0.555	-0.4356	-0.4663
UlcerativeColitis_GSE6731	-0.3112	-0.2555	-0.1799	-0.1451

Table 2.5: KS D-statistic results comparing the DSE distribution against several common distributions. The last column (tS) shows the results against the t distribution after first standardizing DSE by dividing each DSE by the standard deviation of all DSEs.

Dataset	Distribution				
	Normal	Chi-square	F	t	tS
Allergic Rhinitis	0.3109	1	1	0.4991	0.3526
Asthma_GSE4302	0.2795	1	1	0.4895	0.3117
BreastCancer_GSE10810	0.2115	1	1	0.4797	0.3944
CLL_GSE8835	0.1506	1	0.9975	0.4519	0.1596
ColorectalCancer_GSE9348	0.1232	1	0.9994	0.4514	0.2142
CrohnsDisease_GSE6731	0.2131	1	0.987	0.4691	0.2392
LungAdenocarcinoma_GSE7670	0.19	1	0.9999	0.4663	0.332
MS_GDS3920	0.2703	1	0.9994	0.4813	0.3397
Obesity_GSE12050	0.2352	1	0.9991	0.484	0.287
Pancreas_GDS4102	0.2606	1	0.9937	0.4532	0.3254
ParkinsonsDisease_GSE20141	0.0628	1	0.9361	0.3816	0.0582
ProstateCancer_GSE6919_GPL8300	0.1575	1	1	0.4739	0.2522
Psoriasis_GSE13355	0.3327	1	0.9999	0.4932	0.4195
Schizophrenia_GSE17612	0.183	1	0.9998	0.4705	0.2138
T2D_GSE20966	0.3271	1	0.9999	0.4936	0.3562
UlcerativeColitis_GSE6731	0.2397	1	0.998	0.4831	0.3608

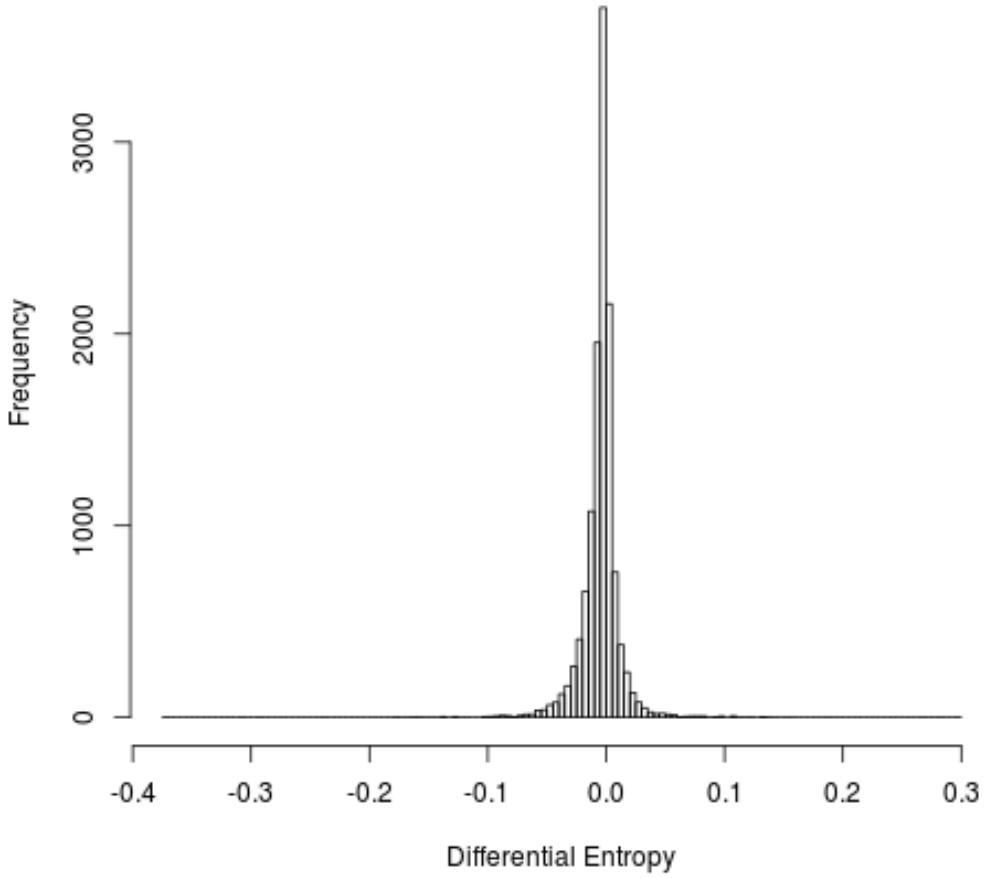

Figure 2.4: The Distribution of Differential Shannon Entropy. The observed distribution of differential Shannon entropy in sample the breastprostate cancer data is shown. Similar patterns were seen in all 16 data sets. None of the standard distributions tested matched the observed distributions closely enough to be considered as a reference distribution for obtaining p-values.

We conclude from this that none of the distributions tested are close enough approximations to the observed DSE distribution to be used as a proxy for obtaining p-values. Thus, without a known distribution function or suitable surrogate, we resort to resampling in order to obtain reliable DSE p-values. While computationally demanding, the following permutation test makes no assumptions about the underlying distribution of the data. Given two lists of numbers, containing n_1 and n_2 numerical elements respectively, we first calculate their DSE and then create a new list A containing all $n_1 + n_2$ numbers from the two lists. Next we randomly permute the elements of A, then recalculate DSE, treating the first n_1 elements of A as one list and the last n_2 elements of A as a second list. The resultant p-value is simply the proportion of all recalculated DSEs that are at least as extreme as the original DSE.

In addition to raw p-values, EntropyExplorer also calculates p-values adjusted for multiple testing. A user can choose to adjust based on FDR, Holm or another multiple-testing adjustment.

2.7 Summary and Future Research Directions

We have studied a pair of alternate techniques for identifying genes of interest in case-control microarray experiments: differential normalised Shannon entropy and differential coefficient of variation. Analysis of 16 human disease datasets has demonstrated that these techniques can identify genes not found by differential expression, and moreover these genes are likely to be disease-relevant based on GO enrichment. Thus, these two techniques can serve as viable alternatives or complements to standard differential expression in an array analysis pipeline. While both techniques are measures of variability, each identifies a sufficient number of genes not found by the other to suggest that they are better used together, and that they complement not only differential expression but also each other.

We have produced EntropyExplorer, an R package for calculating differential Shannon entropy, differential coefficient of variation and differential expression. This package also calculates raw and adjusted p-values for each metric. These measures have been known to complement one another [100], making this package an effective tool for users in search of more expansive suites of differential analysis methods.

In our experiments we found that different datasets may have been processed using different techniques so that some of them look already log-normalized, while others don't. It would be interesting to find out whether different preprocessing techniques would have an impact in identifying related genes, their GO enrichment and the relationship between gene lists selected by different metrics.

Chapter 3

Lower Bounds on Paraclique Density

(Note: This chapter is largely taken from a draft paper under review.)

The scientific literature teems with clique-centric clustering strategies. In this chapter we analyze one such method, the paraclique algorithm. Paraclique has found practical utility in a variety of application domains, and has been successfully employed to reduce the effects of noise. Nevertheless, its formal analysis and worst-case guarantees have remained elusive. We address this issue by deriving a series of lower bounds on paraclique densities.

3.1 Introduction

Clique-centric methods have long played an important role in data science and engineering. Classic techniques include algorithms for $\mathcal{NP}$-hard problems such as maximal clique [14] and maximum clique [13]. The availability of high-throughput data has prompted interest in noise-abatement relaxations, most notably k-clique communities [75] (more recently also called clique percolation) and paraclique [99]. These algorithms have been used for biological data clustering, and been

found superior to traditional methods [51]. Although similar in objective, k-clique communities is hampered in practice by its bottom up approach relying on an exhaustive enumeration of maximal cliques. Paraclique, in contrast, applies top down design principles and employs maximum clique, for which there are highly efficient and reasonably scalable algorithms [92], plus viable alternatives based on duality and parameterized complexity [2].

Paraclique can be formulated in a variety of ways. The general idea is to expand a maximum clique by augmenting it with non-clique vertices adjacent to most, but not all, members of the clique. The motivation for deriving dense subgraphs in this fashion is based on the fact that so-called "missing" edges, while relevant, are often lost due to noise, improper thresholding, weak experimental design, and numerous other causes. A classic example of this phenomenon can be found in the use of DNA microarrays for transcriptomic data analysis. In this setting, vertices represent genes, edges signify co-expression, and paracliques denote molecular response networks differentially (in)activated by stimulus [99]. Depending on a variety of factors, most but not all network elements may be highly intercorrelated at any particular time.

Previous paraclique studies have focused mainly on practical results. Representative examples include [26, 103, 41]. Instead, our primary goal in this chapter is to investigate paraclique's theoretical basis. In so doing, we seek to derive bounds on its worst-case behavior, applying density as the classic clustering metric (we compute a subgraph's density in the traditional way, as the number of edges present divided by the maximum number possible). In the original paraclique formulation, the total number of missing edges was left unchecked. Density could, therefore, in principle be driven to zero. By limiting paraclique size to at most twice the maximum clique size, however, and by requiring that a new non-clique vertex be adjacent to all but one vertex in the growing paraclique, it is known that density is maintained at no less than 50% [99]. Here, we greatly expand upon such density results.

In the next section, we formalize definitions, describe relevant background, and establish several helpful preliminary results. In Sections 3.3 and 3.4, we derive bounds

on general and special cases, respectively. In a final section we draw conclusions and discuss directions for future research.

3.2 Preliminaries

Let G denote a finite, simple, undirected graph. A clique is a subgraph of G in which every pair of vertices is connected by an edge. A paraclique, P, is constructed by first finding a clique C of maximum size, then glomming onto non-clique vertices in a controlled fashion. An integer glom term, g, is used to accomplish this. In the original algorithmic formulation [99], a non-clique vertex was chosen if and only if it was adjacent to g or more vertices in P. The number of required adjacencies does not scale with the size of P using this approach, however, so we generally invert this comparison. Thus, we glom onto a non-clique vertex if and only if it is adjacent to all but at most g vertices in P. In applications, g is usually some small value. In any case we insist that $0 < g < k$, where k denotes the number of vertices in C.

Pseudocode for the paraclique procedure is displayed in Algorithm 1. For the reader's convenience, definitions employed in the sequel are summarized in Table 3.1.

input : graph G, glom term g
output: paraclique P, a subgraph of G

$C \leftarrow$ maximum clique of G
$V \leftarrow$ vertex set of C
while $\overline{V}$ *contains a vertex* v *adjacent to all but at most* g *vertices in* V **do**
  $V \leftarrow V \cup \{v\}$
end
$P \leftarrow$ subgraph induced by V
return P

Algorithm 1: The Paraclique Algorithm

We start by establishing lower and upper bounds on maximum paraclique size.

Lemma 3.1. *A paraclique may contain as many as* $(g+1)k$ *vertices.*

Table 3.1: Definitions used in this chapter.

term	meaning
G	a finite, simple, undirected graph
C	a maximum clique in G
k	the number of vertices in C
P	a paraclique as produced by Algorithm 1
p	the number of vertices in P
g	the glom term used in Algorithm 1
$d(P)$	the density of P

Proof. To construct a paraclique P that satisfies this bound, we begin with $g+1$ disjoint cliques of size k, denoting them $C_0, C_1, \ldots, C_g$, and labeling the vertices of each C_i as $v_{ik}, v_{ik+1}, \ldots, v_{(i+1)k-1}$. To this we add edges connecting vertices v_r and v_s provided they are in different cliques and $r \not\equiv s$ modulo k. The maximum clique size has not changed, since any set of $k+1$ vertices will contain at least two whose indices are in the same equivalance class modulo k (and are thus non-adjacent). Given a graph containing this structure, the paraclique algorithm may return P because v_i is adjacent to all but at most g lower-indexed vertices for any $0 < i \leq (g+1)k - 1$. □

Lemma 3.2. *A paraclique cannot contain more than $2gk$ vertices.*

Proof. Let P denote a paraclique of size $p > k$. By construction, the number of edges in P is at least

$$\begin{aligned}
&\frac{k(k-1)}{2} + (k-g) + (k+1-g) + \cdots + (k+(p-k-1)-g) \\
&= \frac{k(k-1)}{2} + (p-k)(k-g) + (1+2+\cdots+(p-k-1)) \\
&= \frac{k(k-1)}{2} + (p-k)(k-g) + \frac{(p-k-1)(p-k)}{2} \\
&= \frac{p^2 - p - 2pg + 2kg}{2}.
\end{aligned}$$

Since P has no clique of size $k+1$, we know by Turán's Theorem [94] that it contains at most $(1-\frac{1}{k})\frac{p^2}{2}$ edges. Combining the two edge counts produces

$$\begin{aligned}
\frac{p^2-p-2pg+2kg}{2} &\leq \frac{kp^2-p^2}{2k} \\
kp^2-kp-2kpg+2k^2g &\leq kp^2-p^2 \\
p^2-(2g+1)kp+2gk^2 &\leq 0 \\
(p-2gk)(p-k) &\leq 0
\end{aligned}$$

Because $p > k$, we conclude that $p \leq 2gk$. □

3.3 General Case

Let us suppose that C has been isolated, and that g has been chosen. By comparing g and p, we will now prove that as P grows its density approaches 1.0. On the other hand, by comparing g and k, we will also prove that no matter how P changes its density never falls below 0.5.

Theorem 1. *A paraclique's density is at least* $1-\frac{2g-1}{p-1}$.

Proof. As we have previously shown, the number of edges in P is at least

$$\frac{p^2-p-2pg+2kg}{2}.$$

Combining that with Lemma 3.2 ensures

$$\begin{aligned}
d(P) &\geq \frac{\frac{p^2-p-2pg+2kg}{2}}{\frac{p(p-1)}{2}} = \frac{p^2-p-2pg+2kg}{p(p-1)} \\
&\geq \frac{p^2-p-2pg+p}{p(p-1)} = \frac{p-2g}{p-1} = 1-\frac{2g-1}{p-1}.
\end{aligned}$$

□

Theorem 2. *A paraclique's density is at least* $1-\frac{g}{2k-1}$.

Proof. P is missing at most $g(p-k)$ edges. From this it follows that $d(P)$ is bounded below by

$$\frac{\frac{p(p-1)}{2} - g(p-k)}{\frac{p(p-1)}{2}} = 1 - \frac{2g(p-k)}{p(p-1)}.$$

Using basic calculus plus the fact that $p \geq k$, we see that this function takes its minimum on the interval $[k, 2gk]$ at $k + \sqrt{k^2 - k}$. Because $2k - 1 < k + \sqrt{k^2 - k} < 2k$ for all $k \geq 2$, $d(P)$ is minimized when p is either $2k-1$ or $2k$. We know from Lemma 3.1 that both values are possible, and in either case we find that $d(P)$ is at least

$$1 - \frac{2gk}{2k(2k-1)} = 1 - \frac{g}{2k-1}.$$

□

Sometimes Theorem 1 provides the better guarantee. This happens, for example, when $k = 6$, $g = 2$ and $p = 20$. At other times, say when $k = 5$, $g = 4$ and $p = 10$, Theorem 2 provides the tighter result. In any event, the following overall lower bound on density is obtained from Theorem 2 coupled with the fact that $1 \leq g < k$.

Corollary 3. *A paraclique's density always exceeds 1/2.*

3.4 Special Case

The glom term setting $g = 1$ is frequently used in practice. Paraclique structure is considerably more scrutable in this special case. In what follows, we say that P is *nontrivial* if it does not equal C.

Theorem 4. *When $g = 1$, a nontrivial paraclique's density lies between $1 - \frac{2\lfloor \frac{p}{2} \rfloor}{p(p-1)}$ and $1 - \frac{2}{p(p-1)}$, inclusive.*

Proof. Suppose P is such a paraclique. From Lemma 3.2 and the nontriviality of P we know $\lceil \frac{p}{2} \rceil \leq k \leq p - 1$. P must be missing exactly $(p - k)$ edges, else G would

have a clique of size $k+1$. The number of edges in P is thus $\frac{p(p-1)}{2} - (p-k)$, and so

$$d(P) = 1 - \frac{2(p-k)}{p(p-1)} = \frac{2}{p(p-1)}k + \frac{p-3}{p-1}.$$

Given p, this is just a linear function of k with a positive slope. It's minimum therefore occurs when $k = \lceil \frac{p}{2} \rceil$, ensuring

$$d(P) \geq 1 - \frac{2(p - \lceil \frac{p}{2} \rceil)}{p(p-1)} = 1 - \frac{2\lfloor \frac{p}{2} \rfloor}{p(p-1)},$$

and its maximum occurs when $k = p-1$, ensuring

$$d(P) \leq 1 - \frac{2}{p(p-1)}.$$

□

Theorem 5. *When $g = 1$, a nontrivial paraclique's density lies between $1 - \frac{1}{2k-1}$ and $1 - \frac{2}{k(k+1)}$, inclusive.*

Proof. From Lemma 3.2 and the nontriviality of P we know $k+1 \leq p \leq 2k$. Again we note that P must be missing exactly $(p-k)$ edges, and so $d(P) = 1 - \frac{2(p-k)}{p(p-1)}$. As in the proof of Theorem 2, we find from basic calculus that this function is minimized at $1 - \frac{1}{2k-1}$, which occurs at both $p = 2k-1$ and $p = 2k$. It is maximized at $1 - \frac{2}{k(k+1)}$ when $p = k+1$. □

Theorem 4 tends to provide a better lower bound when p is at the lower end of its range relative to k, while Theorem 5 tends to produce a better upper bound when p is at the upper end of its range. In any event, the following overall lower bound on density is obtained from Theorem 5 coupled with the fact that C must contain at least one edge.

Corollary 6. *When $g = 1$, a paraclique's density is always at least 2/3.*

3.5 Conclusions and Directions for Further Research

We have derived density bounds for the paraclique algorithm, a noise-resilient clique-centric technique designed for dense subgraph extraction. To the best of our knowledge, other than the elementary result from [99], these are the first formal density limits for what have come to be popularly known as network community methods. This gives paraclique another potential practical endorsement, in addition to those due to biological enrichment as discussed in [51].

Although we remain primarily concerned with lower bounds, we were able to prove asymptotically tight lower and upper bounds for the special case $g = 1$. Proving better bounds for the general case remains an elusive open problem. Our lower bounds are not tight for arbitrary $g > 1$; our only upper bounds are weak because they are inherited from the special case. If formal analysis proves too difficult, and it may well might, then an alternate approach could center on empirical testing. Both real and synthetic data might be employed to estimate average densities and their expected deviations from our worst-case guarantees.

Chapter 4

Graph Optimization via SAT and ILP: an Empirical Study

SAT and ILP are among the first combinatorial problems shown to be $\mathcal{NP}$-complete. So too are the closely-related graph problems VERTEX COVER, CLIQUE and INDEPENDENT SET. Research over the past few decades has yielded significant algorithmic improvement for each problem. Yet the problems are different enough that research into one does not always readily translate to the others. But being $\mathcal{NP}$-complete problems with known direct polynomial time reductions between them, it is natural to consider whether algorithmic improvements for one problem have outpaced those for another problem to such an extent that performance gain will actually compensate for the overhead of the problem reduction. To this end, in this chapter we consider whether using modern SAT and ILP solvers to find a maximum clique in a graph yields better performance than a fast, direct algorithm. In our tests we compare the MaxSAT solver *akmaxsat*, the ILP solver *CPLEX*, and an in-house implementation of *MCS*, a maximum clique algorithm. Empirical tests show that in most cases a direct clique solver should be the preferred choice, especially on real-world graphs (non-synthetic graphs, or graphs derived from physical or conceptual systems or from empirical observations). But there are instances where SAT and

ILP solvers each outperform other methods. Extreme density and/or regularity are implicated as contributing factors in this behavior. In practice, foreknowledge about graph structure can sometimes be useful in selecting the approach most likely to be fastest, but reliable, general-purpose performance prediction remains an elusive goal.

4.1 Introduction

Reductions are a standard algorithmic method, widely used to show a problem's membership in a particular complexity class. Yet the prevalence and efficacy of reductions as a tool in complexity theory sometimes overshadows the practical potential of transforming one problem into another problem to take advantage of an efficient algorithm for the second problem. Perhaps the most well-known reductions are those between $\mathcal{NP}$-complete problems, many of which are graph problems. The literature contains many examples of solving $\mathcal{NP}$-hard graph problems by using alternate problem formulations. The maximum clique problem, for instance, has been framed in terms of constraint programming [80] and polyhedral methods [6]. Graph coloring has been formulated as the satisfiability problem [96, 97]. So too has Hamiltonian Cycle [98].

In this work, we investigate the use of efficient algorithms for satisfiability and integer linear programming to solve the maximum clique problem. Due to its numerous applications in a wide variety of domains, the maximum clique problem has been extensively studied. Many exact algorithms have been developed [89, 16, 74, 32, 93, 64]. Satisfiability and integer linear programming are also classic, well-studied $\mathcal{NP}$-hard problems [53]. Over the past few decades, sophisticated solvers have been developed for both problems. For example, many solvers for variants of the satisfiability problem are presented each year at competitions [86, 5]. And well-known linear programming packages such as IBM ILOG CPLEX Optimization Studio [48] and Gurobi [49] contain suites of algorithms tailored for integer linear programming. With such advanced tools at hand, the question arises whether formulating the

maximum clique problem as satisfiability or integer linear programming can ever result in a performance gain over solving the problem directly. We seek to answer this question.

CLIQUE is closely related to two other archetypical $\mathcal{NP}$-hard graph problems, INDEPENDENT SET and VERTEX COVER. When framed as decision problems with parameter k, the respective problems ask whether a graph contains a clique, an independent set, or a vertex cover of size k. The relationship between the three problems is summarized as follows. Given a graph G with n vertices and its complement $\overline{G}$, G has a clique of size k if and only if $\overline{G}$ has an independent set of size k, and G has a clique of size k if and only if $\overline{G}$ has a vertex cover of size $n-k$. There are differences between the problems, of course. For instance, VERTEX COVER is the only one of the three that is fixed-parameter tractable (unless $W[1] = FPT$) [24]. But even so, the near-equivalence of the problems suggests that practical findings on one can be extended, at least in part, to the others.

Each problem has a decision version and an optimization version. For instance, the CLIQUE decision problem inputs a graph G and a nonnegative integer k and asks whether G contains a clique with k vertices. Its optimization version, MaxCLIQUE, inputs a graph G and seeks the number of vertices of a clique of maximum size in G. The SAT decision problem inputs a Boolean formula ϕ and asks whether there exists an assignment of variables such that ϕ evaluates to true. Its optimization version, MaxSAT, inputs a Boolean formula ϕ and seeks the maximum number of clauses that can be satisfied by some assignment of variables. The optimization version of ILP inputs an objective function and a set of linear constraints and seeks an assignment of variables that maximizes (or minimizes) the objective function, where variables are restricted to integer values. The decision version of ILP omits the objective function, merely asking whether some point exists which satisfies the constraints. Henceforth, when referring to ILP, we mean the optimization version. In this work, we consider only the optimization versions: MaxCLIQUE, MaxSAT and ILP.

To narrow the scope of our study to a manageable level, we selected one representative solver for each of the three problems: MaxCLIQUE, MaxSAT, and ILP. Other solvers would perform somewhat differently, of course. But our study is not intended to be a systematic comparison of the plethora of tools available for each problem. Testing a fast representative solver for each problem type is enough to establish that some types of problems may be feasibly solved via reductions. The solvers were chosen for availability, demonstrated performance against their peers and usability under Linux. For MaxCLIQUE, we used our own implementation of the *MCS* algorithm from [93]. For MaxSAT, we used *akmaxsat* version 1.1 [57]. And for ILP, we used IBM ILOG CPLEX Optimization Studio version 12.5. We refer to the implementations as *MCS*, *akmaxsat* and *CPLEX*.

Often there are multiple ways to transform one problem into another. We therefore sought in the literature for different reductions from MaxCLIQUE to variants of MaxSAT and ILP. We test a total of five reduction methods, three from MaxCLIQUE to MaxSAT and two from MaxCLIQUE to ILP. We only tested reductions that run in polynomial time. For this reason, we did not consider one of the MaxCLIQUE to Binary ILP reductions in [13], since it involves enumerating all maximal independent sets. The reduction to MaxSAT all result in Boolean formulas in conjunctive normal form (CNF), so changing our definition of MaxSAT to require input in CNF would not affect the results. When doing timings, we exclude the runtime of the reduction, noting that it is a tiny fraction of the runtime anyway except in the most trivial instances.

The rest of this chapter is organized as follows. Section 2 gives three MaxCLIQUE to MaxSAT reductions. Section 3 describes two MaxCLIQUE to ILP reductions. Section 4 describes our methodology, each of the solvers tested, the test graphs, the testing environment and the timing methods. Section 5 gives results of performance trials using *MCS*, *akmaxsat* and *CPLEX* on the test graphs, using the different reduction methods. And finally, section 6 provides a concluding discussion and directions for future research.

4.2 MaxCLIQUE to MaxSAT Reductions

We tested three reductions of MaxCLIQUE to variants of MaxSAT. One reduction was to Max2SAT; the other two reductions were to partial MaxSAT. The solver we tested, *akmaxsat*, contains algorithms for both variants. MaxSAT seeks the maximum number of clauses in a CNF Boolean formula ϕ that can be simultaneously satisfied by some truth assignment of variables. Max2SAT restricts the clauses to two literals each (2CNF). Whereas 2SAT (the problem of determining whether a given 2CNF formula is satisfiable) is in $\mathcal{P}$, Max2SAT (the problem of determinining the maximum number of simultaneously satisfiable clauses in a 2CNF formula) is $\mathcal{NP}$-hard [37]. A MaxSAT instance is *partial* if some prespecified subset of clauses are *hard*, meaning they must be satisfied in every assignment. The remaining clauses are *soft* and can be either satisfied or unsatisfied. An assignment that satisfies all hard clauses is called *feasible*, no matter how many soft clauses it satisifies. The partial MaxSAT problem seeks the maximum number of soft clauses that can be satisfied while satisfying all the hard clauses. We tested one reduction from MaxCLIQUE to Max2SAT and two reductions from MaxCLIQUE to partial MaxSAT.

4.2.1 MaxCLIQUE to Max2SAT Reduction

This reduction is from [90]. Given a graph $G = (V, E)$ with n vertices, it produces a 2CNF formula ϕ with n variables corresponding to the vertices of G, denoted by $x_1, x_2, ..., x_n$, and an additional dummy variable z. It constructs the following clauses.

1. Two clauses $(x_i \vee z)$ and $(x_i \vee \neg z)$ for each variable x_i.

2. One clause $(\neg x_i \vee \neg x_j)$ for each non-edge $(i, j) \notin E$.

Variable x_i is assigned to true in any optimum assignment if and only if it is in some maximum clique of G. There are $2n$ type 1 clauses and $\binom{n}{2} - |E|$ type 2 clauses, for a total of $2n + \binom{n}{2} - |E|$ clauses. The graph G has a k-clique if and only if ϕ has

a truth assignment that satisfies $n + \binom{n}{2} - |E| + k$ clauses. So the maximum clique size of G is obtained by subtracting $n + \binom{n}{2} - |E|$ from the solution of the Max2SAT instance.

4.2.2 MaxCLIQUE to Partial MaxSAT Reduction: Method 1

The first MaxCLIQUE to partial MaxSAT reduction is from [64]. For a graph $G = (V, E)$ with n vertices, it produces a partial MaxSAT instance ϕ with n variables corresponding to the vertices of G, denoted $x_1, x_2, ..., x_n$. Using these variables, it constructs the following clauses.

1. A hard clause $(\neg x_i \vee \neg x_j)$ for each non-edge $(i, j) \notin E$.
2. A soft clause for each vertex in G, each clause containing a single literal x_i.

The hard clauses guarantee that in any feasible assignment of ϕ, the set of true variables corresponds to set of vertices comprising a clique in G. Any non-edges between vertices corresponding to true variables would otherwise result in an unsatisfied hard clause. The number of soft clauses satisified by a feasible assignment is precisely the number of variables set to true, therefore the maximum number of satisfiable soft clauses is exactly the size of a maximum clique in G. In such an optimum assignment, x_i will be true if and only if its corresponding vertex in G is in a maximum clique.

This reduction produces $\binom{n}{2} - |E|$ hard clauses with two literals each and n soft clauses with one literal each (unit clauses).

4.2.3 MaxCLIQUE to Partial MaxSAT Reduction: Method 2

Also from [64], the second MaxCLIQUE to partial MaxSAT reduction is called *independent set based MaxSAT encoding*. Given a graph $G = (V, E)$ with n vertices, it

first partitions G into independent sets, i.e. assigns colors to the vertices of G so that no two adjacent vertices have the same color. The partition need not be optimal, i.e. use the minimum number of colors. Greedy coloring can perform this step in polynomial time.

Once it has a partition of G, this method, like method 1, produces a partial MaxSAT instance ϕ with n variables corresponding to the vertices of G, denoted $x_1, x_2, ..., x_n$. It constructs the following clauses.

1. A hard clause $(\neg x_i \vee \neg x_j)$ for each non-edge $(i, j) \notin E$.

2. A soft clause $(x_i \vee ... \vee x_j)$ for each independent set (color class) in the partition of G into independent sets, where variables $\{x_i, ..., x_j\}$ correspond to vertices in one independent set.

Method 1 can be viewed as a special case of method 2, with size one independent sets. As in method 1, the hard clauses guarantee that true variables in a feasible assignnment of ϕ correspond to a clique in G. The hard clauses also guarantee that no soft clause can have more than one variable assigned to true, since variables in each soft clause represent an independent set in G. Therefore, the maximum number of satisfiable soft clauses in ϕ is precisely the size of a maximum clique in G.

One advantage of this method is that (except in the trivial case of a complete graph) it results in fewer soft clauses than method 1. It produces the same $\binom{n}{2} - |E|$ hard clauses with two literals each as method 1, and $O(n)$ soft clauses with a total of n literals.

4.3 Reductions of MaxCLIQUE to ILP

We tested two reductions from MaxCLIQUE to 0-1 ILP, or binary ILP. The binary ILP problem is a special case of ILP where the variables are restricted to values 0 or 1. Binary ILP is $\mathcal{NP}$-hard. ILP is formulated as an objective function and a set of linear constraints. A traditional formulation is as follows:

maximize $\boldsymbol{c}^T\boldsymbol{x}$, *such that* $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}, \boldsymbol{x} \in \mathbb{Z}^n$

Here $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{b} \in \mathbb{R}^m$ are given vectors or matrices with appropriate dimensions as input.

The size of the problem instance produced by each reduction is summarized in Table 4.1.

4.3.1 MaxCLIQUE to ILP Reduction: Method 1

This reduction is from [13]. Given a graph $G = (V, E)$ with n vertices labeled $1, ..., n$, it constructs a binary ILP instance with n binary variables denoted by $x_1, x_2, ..., x_n$, each of which corresponds to a vertex in G. Using these variables, it constructs a 0-1 ILP instance with the following parts.

1. An objective function $x_1 + x_2 + ... + x_n$.

2. A constraint $x_i + x_j \leq 1$ for each non-edge $(i, j) \notin E$.

The constraints guarantee that the set of variables set to 1 correspond to a clique in G. The objective function is maximized when the maximum possible number of variables are set to 1. Thus, the maximum clique size in G is the optimum value of the objective function, and the variables set to 1 in such an optimum assignment correspond to a maximum clique in G.

4.3.2 MaxCLIQUE to ILP Reduction: Method 2

We modify method 1 in a similar fashion to how the MaxCLIQUE to Partial MaxSAT reduction in section 4.2.3 modifies the reduction in section 4.2.2. Similarly, we call this reduction *independent set based ILP encoding*. Given an input $G = (V, E)$ with n vertices, this method first partitions G into χ independent sets. As in section 4.2.3, the partition need not be optimal; it can be accomplished with greedy coloring. Using binary variables $x_1, x_2, ..., x_n$ corresponding to the vertices of G and binary variables

$y_1, y_2, ..., y_\chi$ corresponding to the independent sets in a partition of G, it constructs a Binary ILP instance with the following parts.

1. An objective function $y_1 + y_2 + ... + y_\chi$.
2. A constraint $x_i + x_j \leq 1$ for each non-edge $(i, j) \notin E$.
3. A constraint $x_i + ... + x_j \geq y_k$ for each independent set in a partition of G into independent sets, where variables $\{x_i, ..., x_j\}$ correspond to vertices in the k^{th} independent set.

An advantage of this reduction is that it has a smaller objective function than method 1. As in method 1, the constraints in 2) guarantee that the set of x variables set to 1 correspond to a clique in G. Since each independent set in a partition of G can contribute at most one vertex to any clique, each y variable evaluates to 1 if and only if its corresponding independent set contributes exactly one vertex to the clique. Thus number of y variables set to 1 when the objective function is maximized is equal to the size of a maximum clique in G.

This method produces a Binary ILP instance with $n+\chi$ variables and $\binom{n}{2} - |E| + \chi$ constraints.

Table 4.1: The number of variables and clauses/constraints produced by each of the tested reductions. Note that χ is the number of colors used in a greedy coloring.

Reduction	Variables	Clauses or Constraints
MaxCLIQUE to Max2SAT	$n+1$	$2n + \binom{n}{2} - \lvert E \rvert$
MaxCLIQUE to Partial MaxSAT Method 1	n	$n + \binom{n}{2} - \lvert E \rvert$
MaxCLIQUE to Partial MaxSAT Method 2	n	$\chi + \binom{n}{2} - \lvert E \rvert$
MaxCLIQUE to ILP Method 1	n	$\binom{n}{2} - \lvert E \rvert$
MaxCLIQUE to ILP Method 2	$n+\chi$	$\chi + \binom{n}{2} - \lvert E \rvert$

4.4 Testing Methodology

We selected and tested one solver to act as representative for each of three problems, MaxCLIQUE, MaxSAT and ILP. Our goal was a first pass at testing whether reducing

graph problems to SAT and ILP provides any practical advantage. A systematic survey and test of existing algorithms for each problem is thus well beyond our scope. It is often observed that there is no single "best" algorithm for a given $\mathcal{NP}$-complete problem that outperforms all other algorithms on every problem instance. We make no claim that our selections are in any sense the "best" algorithms for a given problem, only that each is a reasonable choice as a representative algorithm for its problem. The algorithms were selected for availability, demonstrated competitive performance against their peers, and usability under Linux. We only selected sequential code, or code that could be restricted to a single thread.

For MaxCLIQUE, we tested our own C implementation of the *MCS* algorithm from [93]. For MaxSAT, we used *akmaxsat* version 1.1 [57]. And for ILP, we used IBM ILOG CPLEX Optimization Studio version 12.5 [48]. We refer to the implementations as *MCS*, *akmaxsat* and *CPLEX*.

The MaxSAT solver *akmaxsat* is designed to call the local search solver *ubcsat* if it is available. Based on our experiences, some local search solvers, including *ubcsat*, can be very fast at finding good heuristic solutions, possibly much faster than most exact solution methods can find the same solution. We think it may not be fair to let *akmaxsat* call *ubcsat* before it starts its exact search. Thus we have compiled a version of *akmaxsat* that does not call the stochastic local search solver *ubcsat*.

CPLEX is a sophisticated solver with numerous configurable runtime options, including algorithmic choices that affect the shape of the search tree. By default, it takes advantage of multithreading. To test it as a true sequential solver, we configured it to use only one thread. The only other parameter we changed during our experiments is the branching strategy for mixed integer programming. The default for this parameter is "best-bound search." On some problem instances, this choice caused an out-of-memory error, apparently because it stores a large part of the search tree in memory. When this occurred, we changed this parameter to "depth-first search," which only stores the current path from the root of the search tree. *CPLEX* contains many other parameters and algorithmic choices, some of which undoubtably would

perform differently than the default parameters on particular problem instances. For the same reasons we chose a single representative solver for each problem, we only consider the default configuration for *CPLEX*.

4.4.1 Testing Environment

All compilation and timing experiments were conducted on the Ubuntu 12.04 (Precise Pangolin) x64 Linux operating system running on a Dell OptiPlex 9010 Minitower with a 3.40GHz Intel Core i7-3770 CPU, 16.0GB of DDR3 non-ECC SDRAM memory at 1600MHz (4 DIMMs) and a 500GB 7200 RPM SATA hard drive at 6Gbps with 16MB cache. Compilation was done using GCC version 4.6.3 compiler. *MCS* and *akmaxsat* were compiled with the -O3 switch.

All reported runtimes are in seconds. Runs that did not finish within 50 hours were terminated. Therefore, any runtime shown as 50 hours (180,000 seconds) in the timings figures means that the run did not finish within 50 hours. The time to perform the problem reduction was not included in the reported runtime. In any event, the reduction time was always a small fraction of the overall runtime for a problem instance.

4.4.2 Test Graphs

We tested a variety of graphs, divided into 5 broad categories: random, regular, DIMACS, BHOSLIB and real-world. The random graphs were generated according to the Erdös-Rényi model [30], where each edge has equal probability of being present, independent of other edges. The probability is the desired density of the graph, where density is the proportion of possible edges actually present. We tested random graph sizes of 200, 230 and 250 vertices, at densities ranging from 10% to 99%. The sizes were selected so that a majority of the random graphs would finish on most of the solvers.

We obtained regular graphs from a number of sources, including a generator that produces graphs with a given degree sequence from [21]. The four *usr* series regular graphs are from [77]. They are constructed using strongly regular graphs as basic components, connecting each vertex of each component to all the vertices of all the other components. The 188-regular graph *reg_203_188* with 203 vertices was created by the generator from [21] to have the same degree sequence as *usr7_203-1* and *usr7_203-2* from the *usr* series. See table 4.2 for properties of the tested regular graphs. Note that there are a huge number of d-regular graphs with n vertices (when nd is even) for reasonably large n and d based on a conjectured formula [70] to calculate the asymptotic number of non-isomorphic unlabeled graphs with a given number of vertices regular of a given degree.

Table 4.2: Regular Graphs used in section 4.5.2

graph	vertices	edges	$\omega(G)$
usr4_116-1	116	5858	20
usr4_116-2	116	5858	20
usr7_203-1	203	19082	35
usr7_203-2	203	19082	34
reg_203_188	203	19082	46

DIMACS [52] and BHOSLIB [104] both provide online repositories of synthetic graphs frequently used to benchmark maximum clique solvers. The former provides graphs generated using a variety of mathematical methods. The latter provides graphs with hidden solutions. See table 4.3 for properties of the tested DIMACS graphs and table 4.4 for properties of the tested BHOSLIB graphs.

We tested real-world graphs representing a wide variety of domains, including biology, transportation, communications, social networks, and word association. We obtained the graphs from two major repositories, the Stanford Large Network Dataset Collection (SNAP) [62] and the Koblenz Network Collection (KONECT) [59]. Several of the transcriptomic graphs were constructed using datasets obtained from the Gene Expression Omnibus (GEO) [28]. Others were created from in-house transcriptomic data. See table 4.5 for properties of the tested real-world graphs..

Table 4.3: DIMACS Graphs used in section 4.5.3

graph	vertices	edges	$\omega(G)$
Brock200_1	200	14834	21
Brock200_2	200	9876	12
Brock200_3	200	12048	15
Brock200_4	200	13089	17
gen200_p0.9_44	200	17910	44
gen200_p0.9_55	200	17910	55
hamming8-2	256	31616	128
hamming8-4	256	20864	16
MANN_a27	378	70551	126
MANN_a45	1035	533115	345
johnson16-2-4	120	5460	8
johnson32-2-4	496	107880	16
p_hat300-1	300	10933	8
p_hat300-2	300	21928	25
p_hat300-3	300	33390	36
san200_0.7_1	200	13930	30
san200_0.7_2	200	13930	18
san200_0.9_1	200	17910	70
san200_0.9_2	200	17910	60
sanr200_0.7	200	13868	18
sanr200_0.9	200	17863	42

We tested other graphs besides those reported. We omitted those that were solved nearly instantly (in less than a second) by all the solvers, since they are not informative about the difference between the methods. Similarly, we omitted results for any graphs that were not solved by at least one method within 50 hours.

For each graph, we report six runtimes: one for *MCS* and one for each of the five reductions given in sections 4.2 and 4.3, each of which makes use of the appropriate solver, either *akmaxsat* or *CPLEX*, on the resulting problem instance. These runtimes are labeled as *MCS*, *Max2SAT*, *PMaxSAT-1*, *PMaxSAT-2*, *ILP-1* and *ILP-2*.

Table 4.4: BHOSLIB Graphs used in section 4.5.2

graph	vertices	edges	$\omega(G)$
frb30-15-1	450	83198	30
frb30-15-2	450	83151	30
frb30-15-3	450	83216	30
frb30-15-4	450	83194	30
frb30-15-5	450	83231	30

4.5 Results and Discussion

The first goal of our experiments was to determine if a MaxSAT or ILP solver could ever outperform a well-tuned maximum clique solver on a maximum clique problem instance. We had no particular hypothesis at the outset, although experience suggests that it ought to be difficult to outperform an algorithm tailored for a particular problem with one designed for a different problem. Perhaps the most uninteresting result would have been if *MCS* handily outperformed *akmaxsat* and *CPLEX* on all the test graphs. Such was not the case, however.

A number of other questions arise. For instance, how much performance difference is there between instances created with different reductions from the same problem instance? And what types of graphs are amenable to solution by reduction to MaxSAT and/or ILP? The gold standard would be a model to accurately predict the performance of each solver and reduction based on quickly computable graph characteristics. The model could then be consulted in advance to recommend the best performing method. Unfortunately, our results suggest that such a model may prove elusive for any graphs except random graphs. Random graphs are well-behaved enough that predicting their runtime ought to be a straightforward exercise. Random graphs, unfortunately, are among the graphs we are least interested in solving in real-world applications. Predicting the runtime of the other types of graphs seems far more nuanced than simply using simple metrics such as density and number of vertices.

Table 4.5: Real World Graphs used in section 4.5.5; * means the graph is an induced subgraph of the original graph

graph	vertices	edges	$\omega(G)$	source
adjnoun_adjacency_num	112	425	5	KONECT
airport1	1574	17215	56	KONECT
amazon060_dupr_0_649*	650	3739	9	SNAP
arenas-meta_num_dupr	453	2025	9	KONECT
ca-AstroPh_num_bc_1701_2200*	500	3904	50	SNAP
ca-GrQc_dupr_bc_num_0_499*	500	2362	44	SNAP
Email-Enron1_151_1350*	1200	21150	19	SNAP
facebook_combined_1901_2700*	800	30345	69	SNAP
flickrEdges_num_1001_1700*	700	110662	437	KONECT
globin15.dim	972	149473	23	in-house
HepPh1_1_700*	700	36640	239	SNAP
HepTh1_4401_4700*	300	642	30	SNAP
loc-brightkite_dupr_bc_num_0_599*	600	4524	18	KONECT
loc-gowalla_edges_dupr_0_599*	600	5591	18	KONECT
m430c_f_top2000.70	1130	13005	21	in-house
m430c_m_top2000.70	1123	13430	26	in-house
m430c_s_top2000.70	884	10082	23	in-house
powergrid1_4301_4940*	640	684	6	KONECT
ProstateCancer_GSE6919_case.80	996	4422	33	GEO
ProstateCancer_GSE6919_control.80	1038	5400	53	GEO
roadNet.30000limit1_6101_6900*	800	1303	3	SNAP
sociopatterns-infectious	410	2765	16	KONECT
tsbl6.90_2001_2500*	500	20506	56	in-house
yeast_transfac.99_1401_1800*	400	6077	51	[38]
yeast_transfac.97_1101_2500*	1400	24820	76	[38]

4.5.1 Results on Random Graphs

Figures 4.1, 4.2, and 4.3 show the runtimes on random graphs of size 200, 230 and 250 respectively, at different densities. Each figure shows several densities above 0.9 because this is the range in which the MaxSAT and ILP solvers begin to outperform *MCS*. We make the following observations.

1. When the number of vertices is fixed, the runtime of each method increases with density until a threshold density is reached, above which the runtime begins decreasing with density. The density at which this threshold occurs is different for

each graph size and method, but typically occurs around 0.9 density on our test graphs. *MCS* appears to have a somewhat higher threshold density than the other methods.

2. The threshold density roughly coincides with the density at which the MaxSAT and ILP solvers achieve performance on par with *MCS*.

3. At densities near and above the threshold, both the MaxSAT and ILP solvers begin to outperform *MCS*.

4. The MaxSAT solver outperforms the ILP solver on all tested random graph instances, even those on which both MaxSAT and ILP solvers outperform *MCS*.

5. The choice of reduction method makes very little difference. All three MaxSAT reductions exhibit nearly identical performance, as do both ILP reductions.

The threshold density phenomenon bears similarity to the threshold ratio between the number of clauses and variables for the random k-SAT problem [71]. This can be seen if we treat the number of clauses as the number of edges of a graph and the number of variables as the number of vertices of the graph.

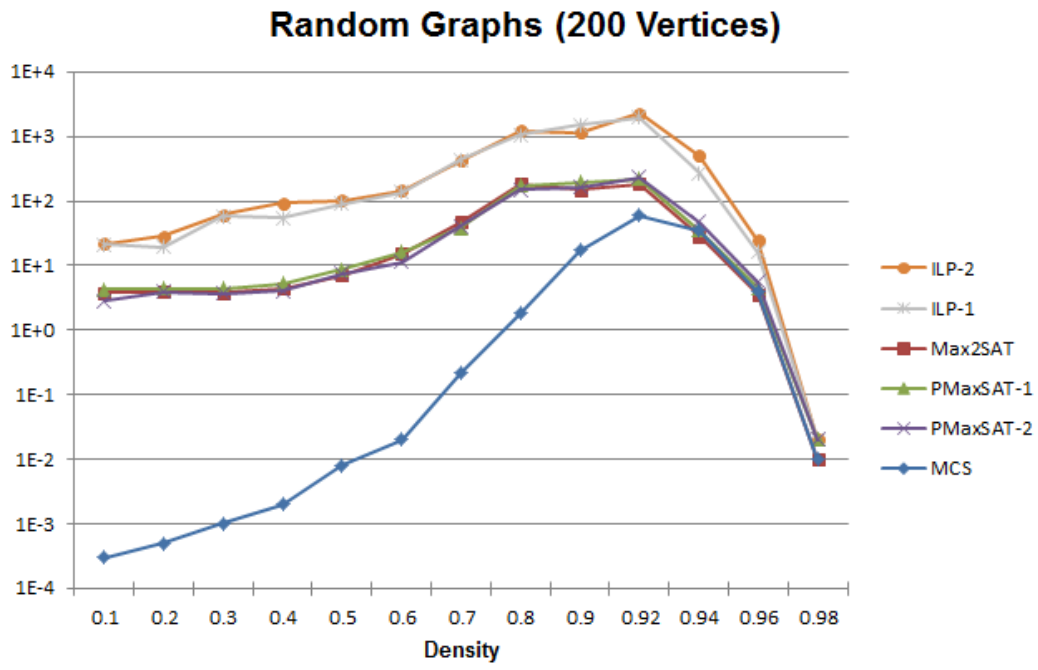

Figure 4.1: Performance on random graphs with 200 vertices at different densities.

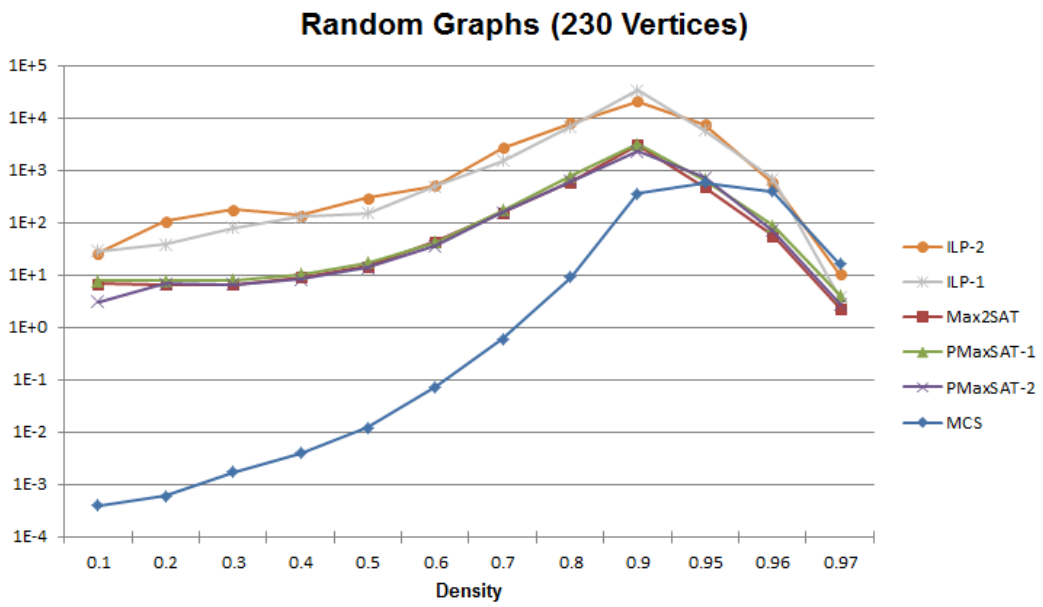

Figure 4.2: Performance on random graphs with 230 vertices at different densities.

4.5.2 Results on Regular Graphs

Figure 4.4 shows the performance of the six methods on the five regular graphs we tested. The difference in the behavior of the solvers is striking. This tells us that even regular graphs can have major structural differences. More broadly, it highlights the fact that degree sequence alone is insufficient to make any conclusions about graph structure. Even more strongly, one could say that degree sequence is only a superficial property of graphs. The following are further observations.

1. *CPLEX* has a huge performance advantage over *MCS* and *akmaxsat* on the *usr* graphs. Delving further, we found that *CPLEX* solved the *usr* problem instance during preprocessing, without ever resorting to its "branch and cut" search. We often observe such behavior during the kernelization (preprocessing) phase on fixed-parameter tractable problems such as vertex cover [42]. Cutting planes are a well-known technique in ILP [68]. Based on its output messages, it appears that the host of cutting techniques employed by *CPLEX* are especially effective on the *usr* graphs,

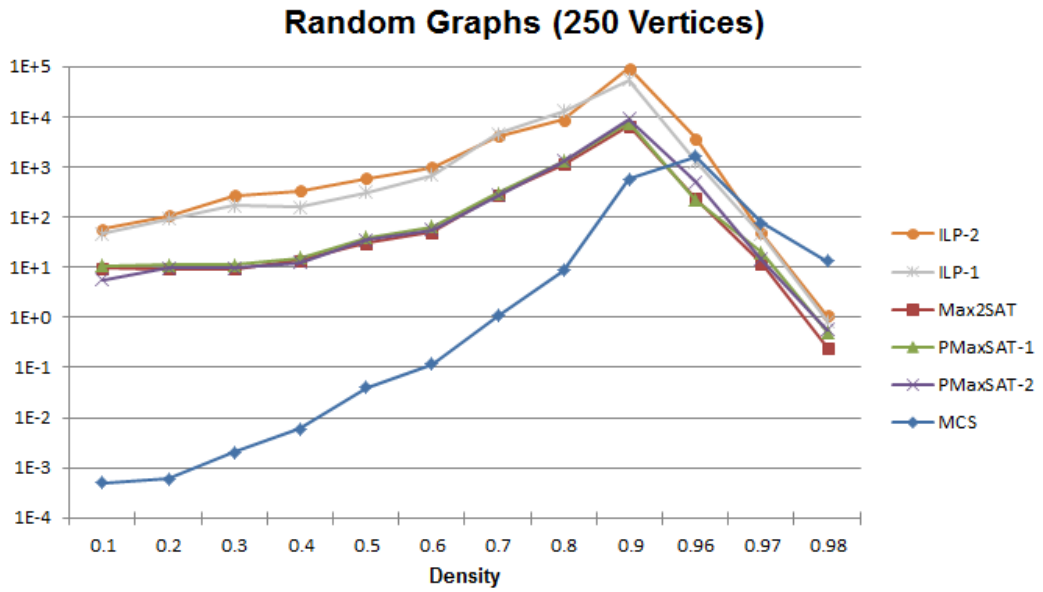

Figure 4.3: Performance on random graphs with 250 vertices at different densities.

i.e. clique cuts, zero-half cuts, Gomory's fractional cuts and lift and project cuts. The effectiveness of cutting techniques does not extend to all regular graphs, however, since *CPLEX* is the slowest of the solvers on *reg_203_188*. Developing graph counterparts for these cutting methods might prove to be a fruitful line of investigation.

2. The MaxSAT solver performs best on *reg_203_188*. Two of its reduction methods, however, do not finish within 50 hours on *usr7_203-2*.

3. Again, the choice of reduction method makes very little difference.

4.5.3 Results on DIMACS Graphs

As noted before, DIMACS graphs are commonly used benchmarks for maximum clique solvers. Because of this, there is a risk that some maximum clique solvers will be specifically tuned for them. Ironically, such a tuning could actually have the unintended side effect of worse performance on graphs other than DIMACS graphs. We mention this issue only in the interest of identifying potential confounds. *MCS* is

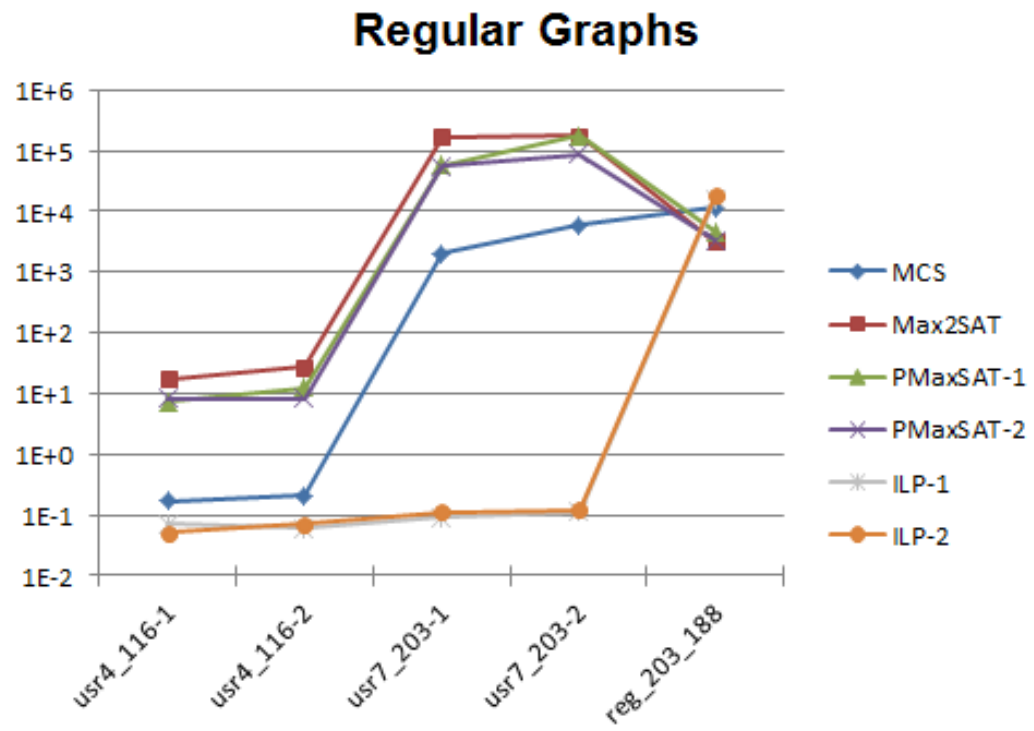

Figure 4.4: Performance on regular graphs.

a general purpose maximum clique algorithm, and we made no attempt to tune our implementation to any particular type of graphs.

There are a wide variety of DIMACS graphs. There is no imposed commonality, other than being empirically difficult (to varying extent) problem instances for maximum clique solvers. Some DIMACS graphs, such as *C125.9* and *C250.9*, are essentially Erdös-Rényi random graphs. We did not include these, since we tested random graphs in section 4.5.1. Figures 4.5 and 4.6 show the runtimes of the solvers on various DIMACS graphs. We make the following observations. We also comment on the results on the instance *MANN_a81* though none of the three solvers can finish within 50 hours on this instance.

1. Some DIMACS graphs have very similar degree distributions to that of random graphs: *Brock* graphs, for instance. The performance of the solvers on these graphs closely parallels that on random graphs, suggesting that they also share structural

similarities with random graphs as well. The same model used to predict performance on random graphs would probably be accurate for these graphs too.

2. *CPLEX* solved *johnson16-2-4* and *johnson32-2-4* much more quickly than the other solvers. In particular, the performance on *johnson32-2-4* stands out, since none of the other solvers could complete within 50 hours, even *MCS*. Johnson graphs are strongly regular graphs, closely related to Hamming graphs. The *usr* series graphs, on which *CPLEX* also excelled as described in the previous section, are themselves merely regular, not strongly so; however they are constructed from strongly regular components. Yet in contrast to the *usr* series graphs, *CPLEX* did not solve the Johnson graphs completely during preprocessing. In fact, judging from its output messages, *CPLEX* did not apply any cutting techniques. Rather, it was able to remove many constraints before branching. A more precise reason for the fast performance, as well as the structural differences between the Johnson and *usr* graphs, will require deeper investigation.

3. For some very dense DIMACS graphs, such as *MANN_a81* (excluded from the figures), none of the three solvers finished. But based on output messages, it is apparent that *MCS*, *akmaxsat* (using the Max2SAT reduction) and *CPLEX* found a maximum clique quickly, but could not verify that no larger clique existed in time. In contrast, *akmaxsat* with the two reductions to Partial MaxSAT failed to find a maximum clique quickly. If the programs had been allowed to run to completion, this might be an instance with a very large difference between reduction methods.

4. *MCS* was fastest on most DIMACS graphs, *CPLEX* on a few, and *akmaxsat* on none.

4.5.4 Results on BHOSLIB Graphs

BHOSLIB graphs are specifically crafted to be hard instances for maximum clique solvers. The degree distribution is very similar to that of random graphs. They are

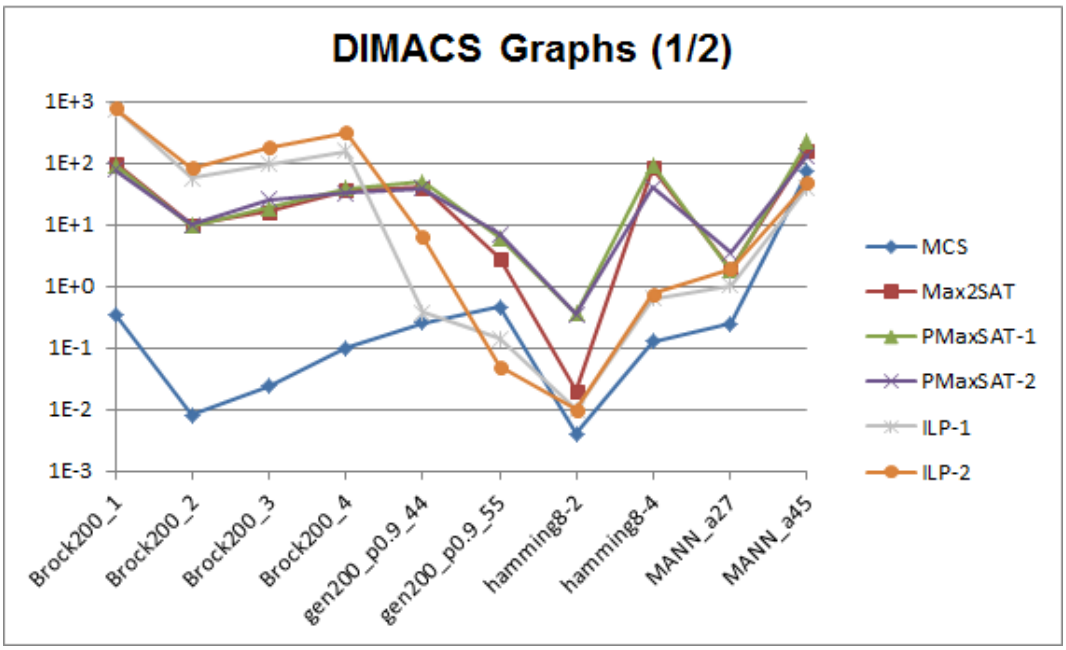

Figure 4.5: Performance on some DIMACS graphs (1).

generated with a more complicated scheme, however, hiding the maximum clique in a clever way [104].

We tested five BHOSLIB graphs, each with 450 vertices. Figure 4.7 shows the results. We observe the following.

1. One of the three MaxSAT reduction methods has starkly different performance than the other two. Max2SAT and PMaxSAT-1 do not finish on any of the graphs. Yet PMaxSAT-2 has the best performance among all the solvers on all five graphs. Why using a soft clause for each independent set rather than a soft clause for each vertex makes such a huge difference on these graphs, but not on any other types of graphs we tested, is a mystery.

2. *CPLEX* and *MCS* are competitive on these graphs, with the edge going to *CPLEX*, since it outperforms *MCS* more often and by a larger margin than graphs on which *MCS* is faster.

3. The two ILP reductions show disparate behavior. ILP-2 outperforms ILP-1 on three of the graphs by as much as an order of magnitude, one graph is a near-tie, and

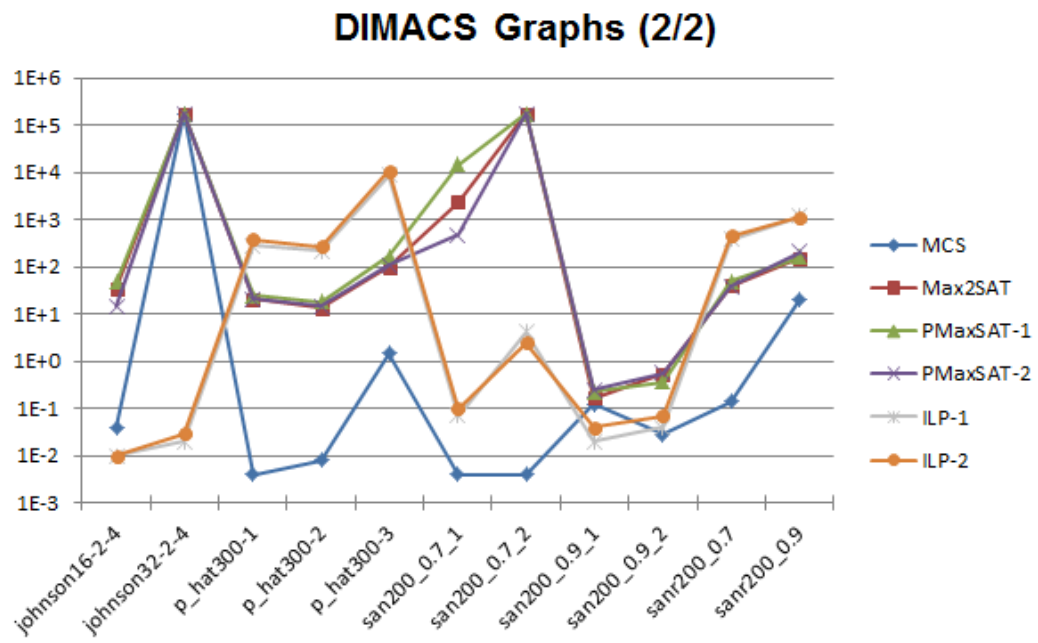

Figure 4.6: Performance on some DIMACS graphs (2).

ILP-1 outperforms ILP-2 on one graph by an order of magnitude. Since the graphs all have 450 vertices and use the same method to hide the maximum clique, the reason for such disparity is elusive.

4.5.5 Results on Real-World Graphs

As mentioned, the 25 real-world graphs we tested came from a variety of domains. See table 4.5 for a listing. In a few cases, we used an induced subgraph of the original graph so that most of the solvers would finish. Such induced subgraphs are indicated by an asterisk (*) in table 4.5.

Figures 4.8 and 4.9 show the results for real-world graphs. We make the following observations.

1. It is immediately clear that *MCS* is the tool of choice for real-world graphs. None of the other methods are competitive, coming no closer than three orders of magnitude on any of the 25 graphs. This is due in large part to the sparsity of such graphs. The reductions to MaxSAT and ILP require a clause/constraint for

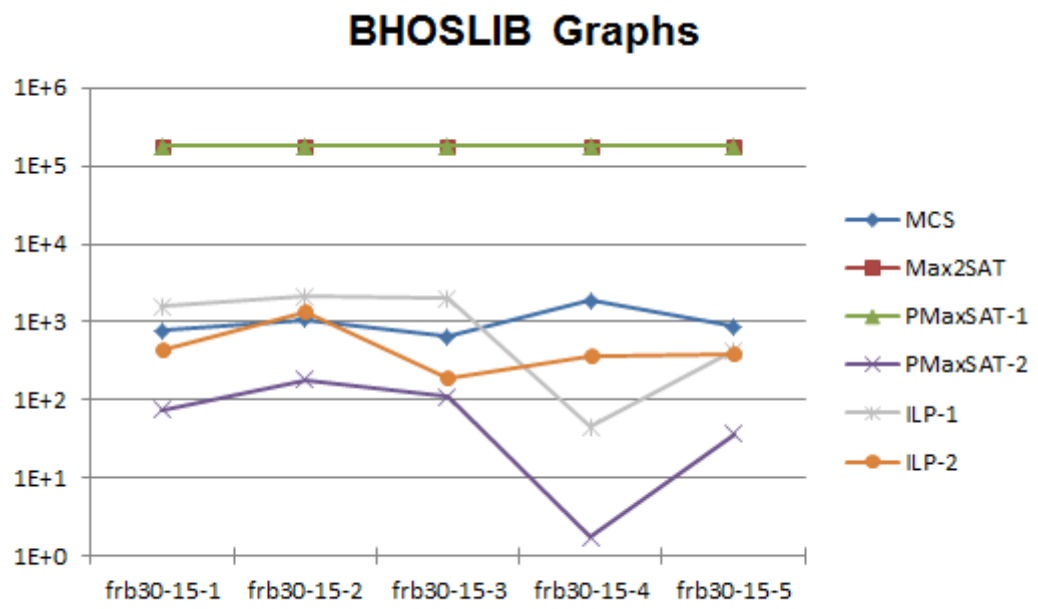

Figure 4.7: Performance on some BHOSLIB graphs.

each non-edge, resulting in very large problem instances when graphs are very sparse. Moreover, the low degree of most vertices in such graphs means that once maximum clique algorithms such as *MCS* find a fairly large clique, they are able to prune huge swaths from the graph, and thus from the search tree.

2. Among the reduction methods, ILP-2 performs well on most of the transcriptomic graphs. But PMaxSAT-2 performs best on nearly all the other graphs. This is evidence of some structural difference between transcriptomic graphs and other types of real-world graphs, as well as some structural similarities between the other types of graphs, even though they are from different domains.

3. Max2SAT and PMaxSAT-1 have nearly identical performance on all the real-world graphs. Moreover, they perform nearly identically on all the graphs we tested.

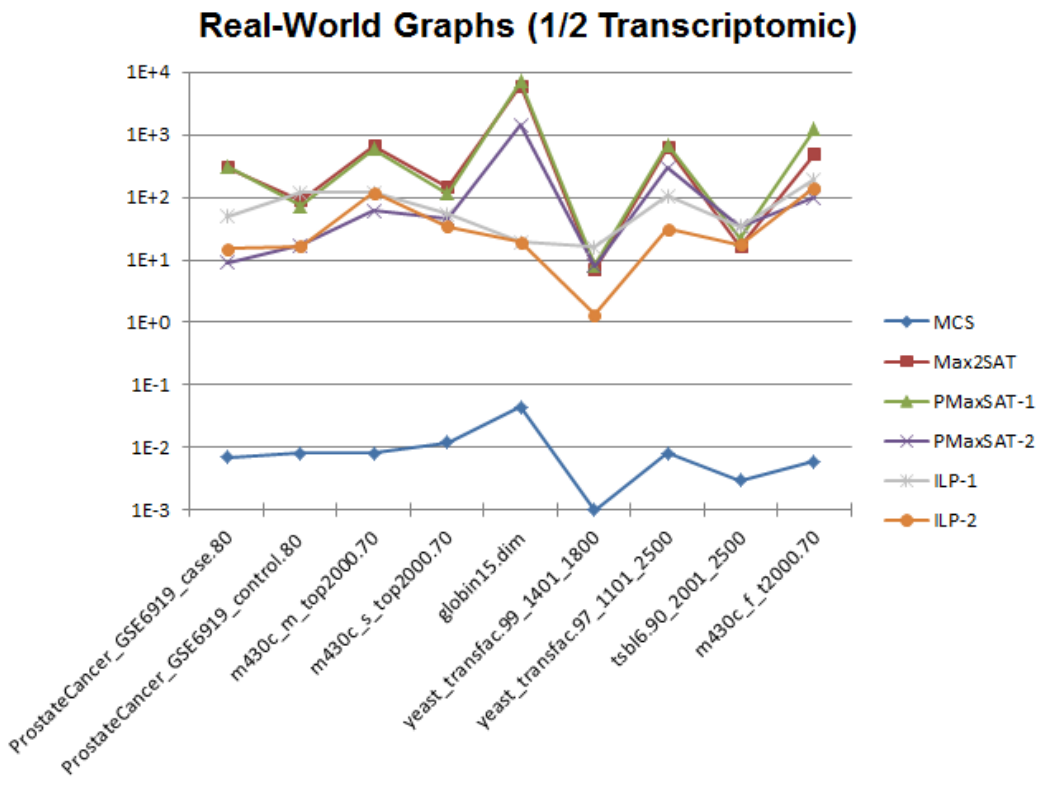

Figure 4.8: Performance on real-world transcriptomic graphs.

4.6 Conclusions and Directions for Future Research

We investigated whether modern implementations of satisfiability and integer linear programming could play an effective role in solving instances of the maximum clique problem. For each of the three problems, we selected one solver to test on a suite of graphs, using five different problem reductions.

Although the direct maximum clique algorithm is usually the fastest, especially on real-world graphs, the MaxSAT and ILP approaches each have their strengths, most notably on very dense random graphs and certain regular graphs. The particular structure that makes such graphs amenable to solution by such techniques appears at least partly related to the preprocessing methods they employ.

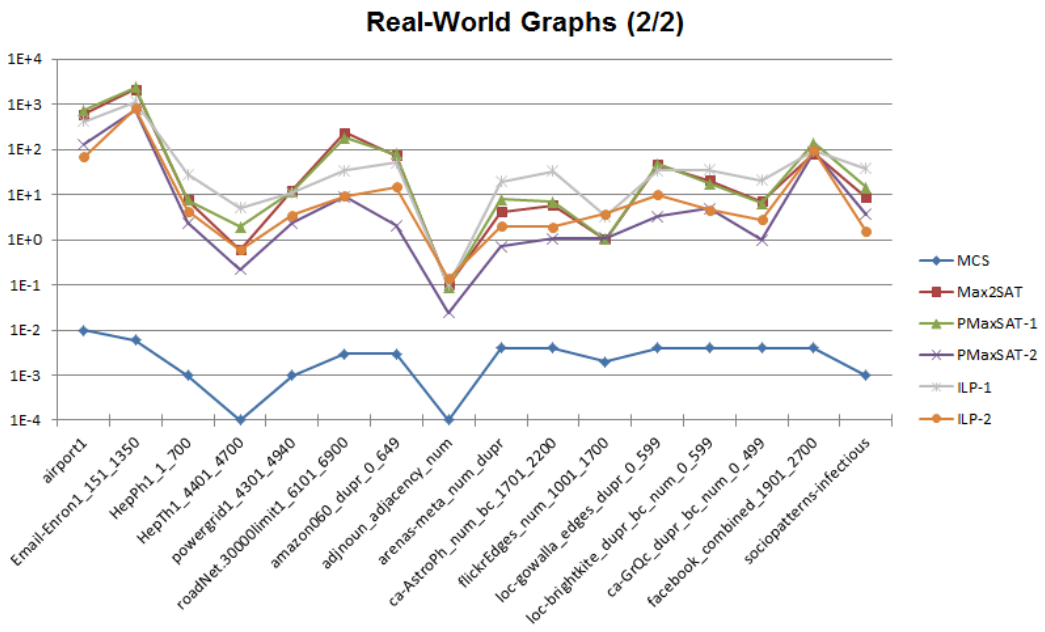

Figure 4.9: Performance on real world graphs.

The research in this chapter suggests many other lines of inquiry. Perhaps the most obvious is to reverse the study, testing whether *MCS* is ever effective at solving instances of MaxSAT and ILP, using reductions mapping those problems to the MaxCLIQUE problem. Another direction is to investigate whether there are reduction methods that yield more tractable formulations of MaxSAT or ILP. We do not purport to have provided an exhaustive survey of known reductions from MaxCLIQUE to MaxSAT and ILP. And of course there may be yet undiscovered reductions on which MaxSAT and/or ILP perform much better than those tested here. A third direction is to study the decision version of the problems. We limited the scope of this present work to the optimization version, the reductions to the decision version are different enough that they may yield different results. See [34, 55, 58, 50] for examples. One could also perform a similar study on related graph problems such as VERTEX COVER and INDEPENDENT SET, each of which has its own algorithms. More generally, since all $\mathcal{NP}$-complete problems are mutually reducible, the same type of experiments could be conducted on any set of $\mathcal{NP}$-complete problems, even going

through multiple problem reductions when a direct reduction between problems is not known.

A deeper investigation of the structural characteristics of graphs that make one solver perform better than others would also be illuminating. Developing graph counterparts to the cutting methods for ILP seem like an especially fruitful avenue for further research, since such cutting methods result in so large a performance gain on particular types of regular graphs.

Chapter 5

Summary and Discussion

In this dissertation we introduced and evaluated two alternative differential metrics for transcriptomic data analysis in identifying disease related genes. The two metrics are shown to be effective augmentations to differential expression in finding highly GO enriched genes. They are also found to complement each other. We also presented how an R package is developed to calculate the differential values and raw and adjusted p-values for the three differential metrics. In this research we found that the ordering of the top ranked genes by differential value and differential p-value can be quite different. It would be interesting to investigate how the differential values and p-values are correlated and why. In the permutation testing for DSE p-value calculation, we found that differential p-values can vary in a non-negligible range if the number of permutations is not large enough with respect to the case and control sample sizes. It would also be interesting to investigate what is the necessary number of permutations to stabilize the p-value for the test.

We reviewed various definitions of paracliques used in the past and analyzed the worst case behavior of an alternate form of paraclique in terms of its densities. We proved various lower bounds of paraclique densities for both the special and the general glom terms. In this way we have shown the performance guarantees of paracliques for its noise abatement clustering role in biological data analysis. The

derived bounds for general glom terms are not tight yet and these could be our future research directions. We could also conduct experiments on real data and find out the expected densities of paracliques under various conditions in comparison to the theoretical bounds.

We presented and evaluated comprehensively various reductions of the clique problem into MaxSAT/ILP problems with the two solvers *akmaxsat* and *CPLEX* in comparison to Tomita's direct *MCS* algorithm for the maximum clique problem. It is shown that although *MCS* is preferable most of the time, the MaxSAT and ILP approaches can sometimes beat the performance of *MCS* and we believe these two approaches can be used to help attack many other $\mathcal{NP}$-hard problems as well. In the future we could fine-tune the classes of graphs for which certain solvers together with the appropriate reductions can be superior to other solving methods. We are aware that the tested graphs in our experiments are at most medium sized. We could perform experiments on larger graphs towards the direction of big data and analyze the scalability of our methods. Moreover, we could also compare performances of direct maximum clique solvers with vertex cover and independent set based methods in addition to the MaxSAT/ILP based methods and identify the regions of input space where each of the methods is most likely to be the best. For every instance for which an indirect method beats a direct algorithm, it is most insightful to find out the reason and if the relevant advantage could be transferred combinatorially into the direct algorithm. Finally, we only considered reductions between optimization versions of the problems in this dissertation. We can also investigate reductions between decision versions of those problems and compare performances.

Bibliography

[1] Faisal N. Abu-khzam, Nicole E. Baldwin, Michael A. Langston, and Nagiza F. Samatova. On the relative efficiency of maximal clique enumeration algorithms, with application to high-throughput. In *Computational Biology, Proceedings, International Conference on Research Trends in Science and Technology*, 2005. 8

[2] Faisal N. Abu-Khzam, Michael A. Langston, Pushkar Shanbhag, and Christopher T. Symons. Scalable parallel algorithms for fpt problems. *Algorithmica*, 45(3):269–284, 2006. 38

[3] D.B. Allison, X. Cui, G.P. Page, and Sabripour M. Microarray data analysis: from disarray to consolidation and consensus. *Nat Rev Genet*, 7(1):55–65, 2006. 7

[4] Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11(10):R106, 2010. 14

[5] Josep Argelich, Chu Min Li, Felip Manya, and Jordi Planes. The maxsat 2006 competition. http://www2.iiia.csic.es/conferences/maxsat06/, 2006. Accessed: 2014-11-03. 46

[6] E. Balas, S. Ceria, G. Cornuéjols, and G. Pataki. Polyhedral methods for the maximum clique problem. *Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993*, page 11, 1996. 46

[7] Arthur G. Bedeian and Kevin W. Mossholder. On the use of the coefficient of variation as a measure of diversity. *Organizational Research Methods*, 3(3):285–297, 2000. 15, 26

[8] T. Beissbarth and T.P. Speed. Gostat: find statistically overrepresented gene ontologies within a group of genes. *Bioinformatics*, 20(9):1464–5, 2004. 11

[9] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of computational biology*, 6(3-4):281–97, 1999. 8

[10] Regina Berretta and Pablo Moscato. Cancer biomarker discovery: The entropic hallmark. *PLoS ONE*, 5(8):e12262, 08 2010. 15, 26

[11] Gabriel F. Berriz, Oliver D. King, Barbara Bryant, Chris Sander, and Frederick P. Roth. Characterizing gene sets with funcassociate. *Bioinformatics*, 19(18):2502–2504, 2003. 11

[12] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981. 7

[13] Immanuel M. Bomze, Marco Budinich, Panos M. Pardalos, and Marcello Pelillo. The maximum clique problem, 1999. 37, 48, 52

[14] Coen Bron and Joep Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, September 1973. 8, 37

[15] Atul J. Butte and Isaac S. Kohane. Relevance networks: A first step toward finding genetic regulatory networks within microarray data. In Giovanni Parmigiani, Elizabeth S. Garrett, Rafael A. Irizarry, and Scott L. Zeger, editors, *The Analysis of Gene Expression Data*, Statistics for Biology and Health, pages 428–446. Springer New York, 2003. 7

[16] Randy Carraghan and Panos M. Pardalos. An exact algorithm for the maximum clique problem. *Oper. Res. Lett.*, 9(6):375–382, November 1990. 46

[17] Bor-Sen Chen and Cheng-Wei Li. On the interplay between entropy and robustness of gene regulatory networks. *Entropy*, 12(5):1071–1101, 2010. 15, 26

[18] Elissa J. Chesler and Michael A. Langston. Combinatorial genetic regulatory network analysis tools for high throughput transcriptomic data. In Eleazar Eskin, Trey Ideker, Ben Raphael, and Christopher Workman, editors, *Systems Biology and Regulatory Genomics*, volume 4023 of *Lecture Notes in Computer Science*, pages 150–165. Springer Berlin Heidelberg, 2006. 8

[19] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2006. 17, 26

[20] Morris H. DeGroot and Mark J. Schervish. *Probability and Statistics*. Pearson, 2011. 30

[21] Charo I. Del Genio, Hyunju Kim, Zoltn Toroczkai, and Kevin E. Bassler. Efficient and exact sampling of simple graphs with given arbitrary degree sequence. *PLoS ONE*, 5(4):e10012, 04 2010. 56

[22] Amira Djebbari and John Quackenbush. Seeded bayesian networks: Constructing genetic networks from microarray data. *BMC Systems Biology*, 2(1):57, 2008. 7

[23] Steven M. Donnelly and Andrew Kramer. Testing for multiple species in fossil samples: An evaluation and comparison of tests for equal relative variation. *American Journal of Physical Anthropology*, 108(4):507–529, 1999. 31

[24] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999. 47

[25] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, 2000. 7

[26] John D. Eblen, Ivan C. Gerling, Arnold M. Saxton, Jian Wu, Jay R. Snoddy, and Michael A. Langston. *Graph Algorithms for Integrated Biological Analysis, with Applications to Type 1 Diabetes Data*, chapter 10, pages 207–222. World-Scientific, 2009. 38

[27] John David Eblen. *The Maximum Clique Problem: Algorithms, Applications, and Implementations*. PhD thesis, University of Tennessee, Knoxville, TN, USA, 2010. 8, 9, 10

[28] Ron Edgar, Michael Domrachev, and Alex E. Lash. Gene expression omnibus: Ncbi gene expression and hybridization array data repository. *Nucleic Acids Research*, 30(1):207–210, 2002. 2, 17, 28, 56

[29] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998. 7, 8

[30] P. Erdös and A. Rényi. On random graphs, I. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959. 55

[31] D.S. Faber and H. Korn. Applicability of the coefficient of variation method for analyzing synaptic plasticity. *Biophysical Journal*, 60(5):1288 – 1294, 1991. 15, 26

[32] Torsten Fahle. Simple and fast: Improving a branch-and-bound algorithm for maximum clique. In *Algorithms ESA 2002*, volume 2461 of *Lecture Notes in Computer Science*, pages 485–498. Springer Berlin Heidelberg, 2002. 46

[33] Michael A. Fligner and Timothy J. Killeen. Distribution-free two-sample tests for scale. *Journal of the American Statistical Association*, 71(353):210–213, 1976. 31

[34] Lance Fortnow. Reductions to sat. http://blog.computationalcomplexity.org/2006/12/reductions-to-sat.html, 2006. Accessed: 2014-07-26. 69

[35] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. *J Comput Biol*, 7(3-4):601–20, 2000. 7

[36] Cesare Furlanello, Maria Serafini, Stefano Merler, and Giuseppe Jurman. Entropy-based gene ranking without selection bias for the predictive classification of microarray data. *BMC Bioinformatics*, 4(1):54, 2003. 15, 26

[37] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1(3):237 – 267, 1976. 49

[38] A.P. Gasch, M. Huang, S. Metzner, D. Botstein, S.J. Elledge, and P.O. Brown. Genomic expression responses to dna-damaging agents and the regulatory role of the yeast atr homolog mec1p. *Molecular biology of the cell*, 12(10):2987–3003, 2001. 59

[39] Audrey P. Gasch and Michael B. Eisen. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biology*, 3(11), 2002. 7

[40] Asghar Ghasemi and Saleh Zahediasl. Normality tests for statistical analysis: A guide for non-statisticians. *International Journal of Endocrinology and Metabolism*, 10(2):486–489, 2012. 31

[41] Thomas Ha, Douglas J. Swanson, Matt Larouche, H. Randy Glenn, Dave Weeden, Peter Zhang, Kristin Hamre, Michael A. Langston, Charles A. Phillips, Mingzhou Song, Zhengyu Ouyang, Elissa J. Chesler, Suman Duvvurru, Roumyana Yordanova, Yan Cui, Kate Campbell, Greg Ricker, Carey R. Phillips, Ramin Homayouni, and Dan A. Goldowitz. Cbgrits: Cerebellar gene regulation in time and space. *Developmental Biology*, 397(1):18–30, 2015. 38

[42] Ronald D. Hagan, Charles A. Phillips, Kai Wang, Gary L. Rogers, Callum Lowcay, and Michael A. Langston. On the relative significance of kernelization versus branching for parallel fpt implementations. In *Proceedings of 11th*

International Conference on Parallel and distributed computing and networks, PDCN 2013. ACTA Press, 2013. 61

[43] Joshua W.K. Ho, Maurizio Stefani, Cristobal G. dos Remedios, and Michael A. Charleston. Differential variability analysis of gene expression and its application to human diseases. *Bioinformatics*, 24(13):i390–i398, 2008. 26

[44] Da Wei Huang, Brad T. Sherman, and Richard A. Lempicki. Systematicand integrative analysis of large gene lists using david bioinformatics resources. *Nature Protocols*, 4(1):44–57, 2008. 10, 17

[45] Da Wei Huang, Brad T. Sherman, and Richard A. Lempicki. Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Research*, 37(1):1–13, 2009. 10, 17

[46] Wolfgang Huber, Anja von Heydebreck, Holger Sltmann, Annemarie Poustka, and Martin Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18(suppl 1):S96–S104, 2002. 14

[47] Curtis Huttenhower, Avi Flamholz, Jessica Landis, Sauhard Sahi, Chad Myers, Kellen Olszewski, Matthew Hibbs, Nathan Siemers, Olga Troyanskaya, and Hilary Coller. Nearest neighbor networks: clustering expression data based on gene neighborhoods. *BMC Bioinformatics*, 8(1):250, 2007. 8

[48] IBM. IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, Last 2010. 46, 54

[49] Gurobi Optimization Inc. Gurobi optimizer reference manual, 2014. 46

[50] Kazuo Iwama and Shuichi Miyazaki. Sat-variable complexity of hard combinatorial problems. In *IN PROCEEDINGS OF THE WORLD COMPUTER CONGRESS OF THE IFIP*, pages 253–258. ELSEVIER SCIENCE B.V, 1994. 69

[51] Jeremy Jay, John Eblen, Yun Zhang, Mikael Benson, Andy Perkins, Arnold Saxton, Brynn Voy, Elissa Chesler, and Michael A. Langston. A systematic comparison of genome-scale clustering algorithms. *BMC Bioinformatics*, 13:S7, 2012. 8, 10, 38, 44

[52] David S. Johnson and Michael A. Trick. *Cliques, Coloring, and Satisfiability*. American Mathematical Society, 1993. 56

[53] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US, 1972. 46

[54] Jyotsna Kasturi, Raj Acharya, and Murali Ramanathan. An information theoretic approach for analyzing temporal patterns of gene expression. *Bioinformatics*, 19(4):449–458, 2003. 7

[55] Massih Khorvash. On uniform sampling of cliques. Master's thesis, University of British Columbia, 2009. 69

[56] I.S. Kohane, A.T. Kho, and A.J. Butte. *Microarrays for an Integrative Genomics*. MIT Press, Cambridge, Mass, 2003. 15, 26

[57] Adrian Kugel. Improved exact solver for the weighted max-sat problem. In Daniel Le Berre, editor, *POS-10*, volume 8 of *EPiC Series*, pages 15–27. EasyChair, 2012. 48, 54

[58] Stefan Kugele. Efficient solving of combinatorial problems using sat solvers. Master's thesis, Technische Universität Mnchen, 2006. 69

[59] Jérôme Kunegis. Konect: The koblenz network collection. In *Proceedings of the 22Nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 1343–1350, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee. 56

[60] Michael A. Langston, Andy D. Perkins, Arnold M. Saxton, Jon A. Scharff, and Brynn H. Voy. Innovative computational methods for transcriptomic data analysis: A case study in the use of fpt for practical algorithm design and implementation. *The Computer Journal*, 51(1):26–38, 2008. 7

[61] J Lapointe, C Li, J.P. Higgins, M van de Rijn, E Bair, K Montgomery, M Ferrari, L Egevad, W Rayford, U Bergerheim, P Ekman, A.M. DeMarzo, R Tibshirani, D Botstein, P.O. Brown, J.D. Brooks, and J.R. Pollack. Gene expression profiling identifies clinically relevant subtypes of prostate cancer. *Proceedings of the National Academy of Sciences of the United States of America*, 101(3):811–6, 2004. 8

[62] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014. 56

[63] Yuk Fai Leung and Duccio Cavalieri. Fundamentals of cdna microarray data analysis. *Trends in Genetics*, 19(11):649 – 659, 2003. 14

[64] Chu Min Li and Zhe Quan. An efficient branch-and-bound algorithm based on maxsat for the maximum clique problem. In *AAAI'10*. AAAI Press, 2010. 46, 50

[65] Shoudan Liang, Stefanie Fuhrman, and Roland Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architectures, 1998. 6

[66] Michael Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome Biology*, 15(12):550, 2014. 31

[67] S. Maere, K. Heymans, and M. Kuiper. Bingo: a cytoscape plugin to assess overrepresentation of gene ontology categories in biological networks. *Bioinformatics*, 21(16):3448–9, 2005. 11

[68] Hugues Marchand, Alexander Martin, Robert Weismantel, and Laurence Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1-3):397 – 446, 2002. 61

[69] L. Masisi, V. Nelwamondo, and T. Marwala. The use of entropy to measure structural diversity. In *Computational Cybernetics, 2008. ICCC 2008. IEEE International Conference on*, pages 41–45, 2008. 15, 26

[70] Brendan D. McKay and Nicholas C. Wormald. Asymptotic enumeration by degree sequence of graphs with degreeso($n^1/2$). *Combinatorica*, 11(4):369–382, 1991. 56

[71] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of sat problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, pages 459–465. AAAI Press, 1992. 60

[72] Sudhir Naswa. *An investigation of gene networks influenced by low dose ionizing radiation using statistical and graph theoretical algorithms*. PhD thesis, University of Tennessee, Knoxville, TN, USA, 2012. 6

[73] Alicia Oshlack, Mark Robinson, and Matthew Young. From rna-seq reads to differential expression results. *Genome Biology*, 11(12):220, 2010. 14

[74] Patric R.J. Ostergard. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120(1-3):197 – 207, 2002. Special Issue devoted to the 6th Twente Workshop on Graphs and Combinatorial Optimization. 46

[75] Gergely Palla, Imre Dernyi, Ills Farkas, and Tams Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, June 2005. 8, 37

[76] Vincente Pedraza, Jose A. Gomez-Capilla, Georgia Escaramis, Carolina Gomez, Pablo Torn, Jose M. Rivera, Angel Gil, Patricia Araque, Nicolas Olea, Xavier

Estivill, and M. Esther Frez-Vidal. Gene expression signatures in breast cancer distinguish phenotype characteristics, histologic subtypes, and tumor invasiveness. *Cancer*, 116(2):486–496, 2010. 28

[77] José Luis López Presa, Antonio Fernández Anta, and Luis Núnez Chiroque. Union of strongly regular graphs. http://pallini.di.uniroma1.it/library/conauto_dim/usr.zip, 2014. Accessed: 2014-07-26. 56

[78] RDevelopmentCoreTeam. *R: a language and environment for statistical computing*. R Foundation for Statistical Computing, 2011. 27

[79] George F. Reed, Freyja Lynn, and Bruce D. Meade. Use of coefficient of variation in assessing variability of quantitative assays. *Clinical and Diagnostic Laboratory Immunology*, 9(6):1235–1239, 2002. 15, 26

[80] Jean-Charles Regin. Using constraint programming to solve the maximum clique problem. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming C CP 2003*, volume 2833 of *Lecture Notes in Computer Science*, pages 634–648. Springer Berlin Heidelberg, 2003. 46

[81] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27:45, 1948. 15, 26

[82] W. Shannon, R. Culverhouse, and J. Duncan. Analyzing microarray data using cluster analysis. *Pharmacogenomics*, 4(1):41–52, 2003. 8

[83] Roded Sharan, Adi Maron-Katz, and Ron Shamir. Click and expander: a system for clustering and visualizing gene expression data. *Bioinformatics*, 19(14):1787–1799, 2003. 8

[84] William B. Sherwin. Entropy and information approaches to genetic diversity and its expression: Genomic geography. *Entropy*, 12(7):1765–1798, 2010. 15, 26

[85] Radha Shyamsundar, Young Kim, John Higgins, Kelli Montgomery, Michelle Jorden, Anand Sethuraman, Matt van de Rijn, David Botstein, Patrick Brown, and Jonathan Pollack. A dna microarray survey of gene expression in normal human tissues. *Genome Biology*, 6(3):R22, 2005. 8

[86] L. Simon, D. Le Berre, and E. Hirsch. The sat 2002 competition. http://www.satcompetition.org/2002/, 2002. Accessed: 2014-11-03. 46

[87] G. K. Smyth. Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, 3:Article3, 2004. 14

[88] Douglas R. Stinson. *Cryptography: Theory and Practice*. Chapman and Hall/CRC, 2006. 23

[89] Robert Endre Tarjan and Anthony E. Trojanowski. Finding a maximum independent set. *SIAM J. Comput.*, 6(3):537–546, 1977. 46

[90] Steven Taschuk. A reduction of clique to max 2-sat. http://www.amotlpaa.org/math/max2sat.pdf, 2007. Accessed: 2014-07-26. 49

[91] S Tavazoie, J.D. Hughes, M.J. Campbell, R.J. Cho, and G.M. Church. Systematic determination of genetic network architecture. *Nat Genet*, 22(3):281–5, 1999. 7

[92] Etsuji Tomita, Tatsuya Akutsu, and Tsutomu Matsunaga. Efficient algorithms for finding maximum and maximal cliques: Effective tools for bioinformatics. In *Biomedical Engineering, Trends in Electronics, Communications and Software*, page Chapter 32, 2011. 38

[93] Etsuji Tomita, Yoichi Sutani, Takanori Higashi, Shinya Takahashi, and Mitsuo Wakatsuki. A simple and faster branch-and-bound algorithm for finding a maximum clique. In Md.Saidur Rahman and Satoshi Fujita, editors, *WALCOM:*

Algorithms and Computation, volume 5942 of *Lecture Notes in Computer Science*, pages 191–203. Springer Berlin Heidelberg, 2010. 46, 48, 54

[94] Paul Turán. Eine extremalaufgabe aus der graphentheorie. *Mat. Fiz. Lapok*, 48(436-452):61, 1941. 41

[95] Virginia Goss Tusher, Robert Tibshirani, and Gilbert Chu. Significance analysis of microarrays applied to the ionizing radiation response. *Proceedings of the National Academy of Sciences of the United States of America*, 98(9):5116–5121, 2001. 31

[96] Allen Van Gelder. Another look at graph coloring via propositional satisfiability. *Discrete Appl. Math.*, 156(2):230–243, January 2008. 46

[97] Miroslav N. Velev. Exploiting hierarchy and structure to efficiently solve graph coloring as sat. In *Proceedings of the 2007 IEEE/ACM International Conference on Computer-aided Design*, ICCAD '07, pages 135–142, Piscataway, NJ, USA, 2007. IEEE Press. 46

[98] Miroslav N. Velev and Ping Gao. Efficient sat techniques for absolute encoding of permutation problems: Application to hamiltonian cycles. In Vadim Bulitko and J. Christopher Beck, editors, *SARA*. AAAI, 2009. 46

[99] Brynn H Voy, Jon A Scharff, Andy D Perkins, Arnold M Saxton, Bhavesh Borate, Elissa J Chesler, Lisa K Branstetter, and Michael A Langston. Extracting gene networks for low-dose radiation using graph theoretical algorithms. *PLoS computational biology*, 2(7):e89, 2006. 7, 37, 38, 39, 44

[100] Kai Wang, Charles A Phillips, Gary L Rogers, Fredrik Barrenas, Mikael Benson, and Michael A Langston. Differential shannon entropy and differential coefficient of variation: Alternatives and augmentations to differential expression in the search for disease-related genes. *International journal of computational biology and drug design*, 7:183–194, 2014. 13, 25, 26, 31, 36

[101] E. U. Weber, S. Shafir, and A. R. Blais. Predicting risk sensitivity in humans and lower animals: risk as variance or coefficient of variation. *Psychological Review*, 111:430–45, 2004. 15, 26

[102] James West, Ginestra Bianconi, Simone Severini, and Andrew E. Teschendorff. Differential network entropy reveals cancer system hallmarks. *Scientific Reports*, 2(802), 2012. 26

[103] Aaron R. Wolen, Charles A. Phillips, Michael A. Langston, Alex H. Putman, Paul J. Vorster, Nathan A. Bruce, Timothy P. York, Robert W. Williams, and Michael F. Miles. Genetic dissection of acute ethanol responsive gene networks in prefrontal cortex: Functional and mechanistic implications. *PLoS ONE*, 7(4):e33575, 04 2012. 38

[104] Ke Xu. Bhoslib: Benchmarks with hidden optimum solutions for graph problems. http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm, 2004. Accessed: 2014-07-26. 56, 65

[105] Bin Zhang and Steve Horvath. A general framework for weighted gene coexpression network analysis. *Statistical applications in genetics and molecular biology*, 4(1), 2005. 8

[106] Yun Zhang, Faisal N. Abu-Khzam, Nicole E. Baldwin, Elissa J. Chesler, Michael A. Langston, and Nagiza F. Samatova. Genome-scale computational approaches to memory-intensive applications in systems biology. In *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, SC '05, page 12. IEEE Computer Society, 2005. 8

Vita

Kai Wang grew up in Changzhou, Jiangsu, China. After finishing high school study in his hometown, he went to the department of Automation of Tsinghua University in Beijing, where he received a Bachelor of Engineering degree in 2000. He went to the department of Computer and Information Science of IUPUI in 2007, where he received a Master of Science degree in Computer Science in 2008. He went to the University of Tennessee in 2009 to begin his doctoral study and joined the Langston Lab in the EECS department in 2011, where he completed his PhD study in 2015.