Efficient Algorithms in Analyzing Genomic Data

Feng Pan

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2009

Approved by:

Wei Wang, Advisor

Leonard McMillan, Co-principal Reader

Fernando Pardo-Manuel de Villena, Reader

David Threadgill, Reader

Jan F. Prins, Reader

Abstract

Feng Pan: Efficient Algorithms in Analyzing Genomic Data.
(Under the direction of Wei Wang.)

With the development of high-throughput and low-cost genotyping technologies, immense data can be cheaply and efficiently produced for various genetic studies. A typical dataset may contain hundreds of samples with millions of genotypes/haplotypes. In order to prevent data analysis from becoming a bottleneck, there is an evident need for fast and efficient analysis methods.

My thesis focuses on two interesting and important genetic analyzing problems.

- Genome-wide Association mapping. The goal of genome wide association mapping is to identify genes or narrow regions in the genome which have significant statistical correlations to the given phenotypes. The discovery of these genes offers the potential for increased understanding of biological processes affecting phenotypes such as body weight and blood pressure.

- Sample selection for maximal Genetic Diversity. Given a large set of samples, it is usually more efficient to first conduct experiments on a small subset. Then the following question arises: What subset to use? There are many experimental scenarios where the ultimate objective is to maintain, or at least maximize, the genetic diversity within relatively small breeding populations.

In my thesis, I developed the following efficient and effective algorithms to address these problems.

- Phylogeny-based Genom-wide association mapping:
  - TreeQA: The algorithm uses local perfect phylogeny tree in genome wide analysis for genotype/phenotype association mapping. Samples are partitioned according to the sub-trees they belong to. The association between a tree and the phenotype is measured by some statistic tests.

  - TreeQA+: TreeQA+ inherits all the advantages of TreeQA. Moreover, it improves TreeQA by incorporating sample correlations into the association study.

- Sample selection for maximal genetic diversity:

  - Sample Selection in biallelic SNP Data: Samples are selected based on their genetic diversity among a set of SNPs. Given a set of samples, the algorithms search for the minimum subset that retains all diversity (or a high percentage of diversity).

  - Representative Sample Selection in Non-Biallelic Data: For more general data (non-biallelic), information-theoretic measurements such as entropy and mutual information are used to measure the diversity of a sample subset. Samples are selected to maximize the original information retained.

Table of Contents

List of Tables

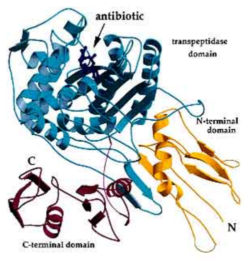
List of Figures

Chapter 1

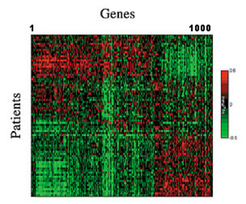
Introduction

Genetics is the study of inheritance and variation in living organisms. A central dogma of modern genetics is that DNA is a template for making RNA which encodes the linear structure of proteins. Thus, biological data such as genomic data and proteomic data are frequently used in modern genetic analysis. In the early years of genetic study, getting sufficient data was the bottle neck of the analysis. Because of the high-cost and low-efficiency of sequencing technologies, scientists only have limited data, e.g. sparse genetic maps. Such data constrain the power of the various genomic analysis. Recently, with the development of high-throughput and low-cost sequencing technologies, immense biological data can now be cheaply and efficiently produced for various genetic studies. A few example datasets are shown in Figure 1.1.

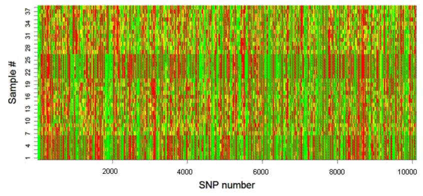
- In Figure 1.1 (a), a protein structural data contains the 3D information for hundreds of amino acid.
- In Figure 1.1 (b), a microarray gene expression data contains thousands of gene expression measurements from different organisms or under different environments.
- In Figure 1.1 (c), a SNP (Single Nucleotide Polymorphism) data contains millions of SNP markers.



(a)



(b)



(c)

Figure 1.1: Examples of immense biological data

These data are broadly used in biological analysis such as Quantitative Traits Loci Analysis (QTL), Gene Regulatory Network, and Protein Family Analysis.

While the immense genomic data are improving the power of various biological analysis, they are also posing great computational challenges to the analysis studies. In QTL mapping, single marker analysis method (Pe'er et al. (2006); Akey et al. (2001)) is fast and frequently used. In a genomic dataset containing millions of markers, one scan of the single marker analysis method already takes hours to finish. Other complex methods such as haplotype-based mapping (McClurg et al. (2006); Onkamo et al. (2002); Wang and Paigen (2005)) and phylogeny-based mapping (Zollner and Pritchard (2005); Mailund et al. (2006); Sevon et al. (2006)) are far more time-consuming. For example, an epistasis test examines all pairs of markers in a genomic data. To run such a test on a genomic data containing 10^6 markers, the total number of tests is $O(10^{12})$. And considering the permutation tests (Fisher (1935)) which are required to calculate the significant threshold, the total number of tests can be further increased by a factor of 10^3 or even more.

Therefore, there is an urgent need to improve the efficiency and scalability of genetic study. The goal of my thesis is to develop efficient analysis methods for various applications in genetics. My thesis focuses on two interesting and important genetic analyzing problems as following:

1. **Phylogeny-based Genome-Wide Association Mapping**: The goal of genome wide association (GWA) mapping in modern genetics is to identify genes or narrow regions in the genome that contribute to genetically complex traits such as morphology or diseases. Among the existing methods, phylogeny-based association mapping methods (Zollner and Pritchard (2005); Mailund et al. (2006); Sevon et al. (2006)) show obvious advantages over single marker-based methods (Pe'er et al. (2006); Akey et al. (2001)) and haplotype-based methods (McClurg et al. (2006); Onkamo et al. (2002); Wang and Paigen (2005)) because they incorporate information about the evolutionary history of the genome into the analysis. However, both the phylogeny inference and the more complex model indicted by the phylogeny cause the existing phylogeny-based methods to be far more time-consuming than single marker and haplotype-based methods. Methods such as TreeLD (Zollner and Pritchard (2005)) can take hours to analyze a small dataset containing tens of samples and markers. Thus, efficient algorithms are in need for genome-wide scale analysis.

2. **Maximum-diversity Sample Selection**: The problem of selecting a sample subset sufficient to preserve genetic diversity, as measured by retaining a specific set of genetic markers, arises in the design of recombinant inbred lines (RIL). RIL panels derived from more than two parental strains, such as the Collaborative Cross(Threadgill et al. (2002); Churchill et al. (2004)), present a particular challenge as to which progenitor strains to include in order to maximize SNP re-

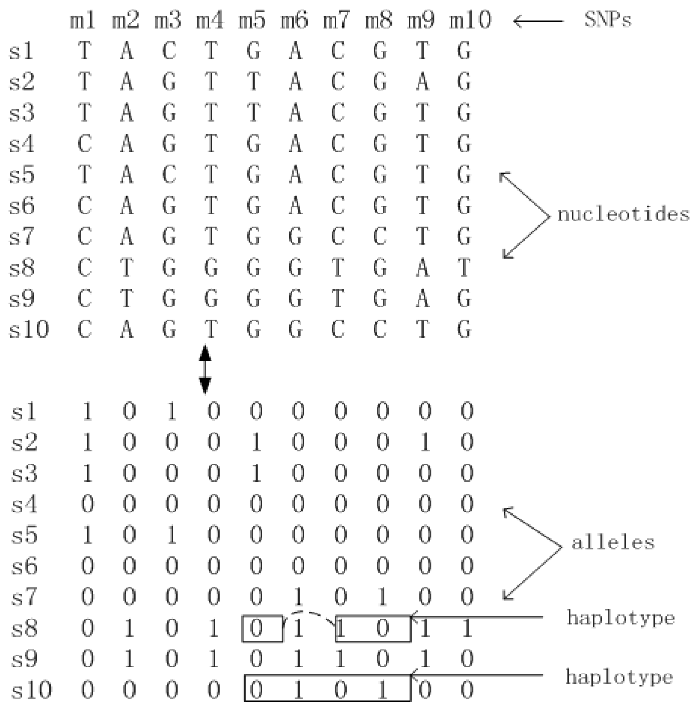
tention. A similar problem occurs when staging association studies across an RIL panel regarding how to order experiments in such a way that the most information is obtained. The problem of finding the sample subset having the greatest diversity is NP-complete. Given n samples, the problem has a searching space of size $O(2^n)$. A typical SNP dataset can contain hundreds of samples which makes it infeasible to perform a manual search. Efficient algorithms which are optimized in runtime by heuristic searching are needed to handle the problem on real data.

In fact, the sample selection problem is closely related to the genome-wide association mapping problem. Genetic (allele) diversity is an important aspect to consider when designing association mapping studies. Sample selection could take place in the following two stages of GWA:

- Design of breeding program: In many cases, association mapping is conducted on a population bred from a small set of samples. Usually, the number of available samples is larger than the number of samples needed to start a breeding program. A subset of samples which has the maximal genetic diversity is usually preferred over a random subset.

- Pre-processing of association mapping: A large set of samples cause computation problems such as the huge number of permutation tests. By selecting a subset of the samples based on their genetic diversity, association mapping can be conducted more efficiently with no significant loss in analysis capability. It also alleviates to some extent the bias caused by population distribution.

While both GWA and sample selection can be conducted on various types of biological datasets, my thesis focuses on the analysis of the genomic data. To be more specific, the genomic data used in my thesis are mostly SNP data produced by isogenic mouse strains. In the isogenic strains, the two copies of each chromosome are identical. Thus, a sample chromosome can be represented by a single string. In these data, Single



	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10
s1	T	A	C	T	G	A	C	G	T	G
s2	T	A	G	T	T	A	C	G	A	G
s3	T	A	G	T	T	A	C	G	T	G
s4	C	A	G	T	G	A	C	G	T	G
s5	T	A	C	T	G	A	C	G	T	G
s6	C	A	G	T	G	A	C	G	T	G
s7	C	A	G	T	G	G	C	C	T	G
s8	C	T	G	G	G	G	T	G	A	T
s9	C	T	G	G	G	G	T	G	A	G
s10	C	A	G	T	G	G	C	C	T	G

s1	1	0	1	0	0	0	0	0	0	0
s2	1	0	0	0	1	0	0	0	1	0
s3	1	0	0	0	1	0	0	0	0	0
s4	0	0	0	0	0	0	0	0	0	0
s5	1	0	1	0	0	0	0	0	0	0
s6	0	0	0	0	0	0	0	0	0	0
s7	0	0	0	0	0	1	0	1	0	0
s8	0	1	0	1	0	1	1	0	1	1
s9	0	1	0	1	0	1	1	0	1	0
s10	0	0	0	0	0	1	0	1	0	0

Figure 1.2: An example SNP data and its corresponding binary matrix

Nucleotide Polymorphisms (SNPs) are used as genetic markers. A SNP is a single nucleotide in the genome which differs between individuals of a species (or between paired chromosomes in an individual). The variants of a SNP are called alleles. Previous work (Ideraabdullah et al. (2004)) has shown that over 99% of SNPs in mouse isogenic strains are biallelic (i.e., have two alleles), which allows us to represent allele diversity as a binary matrix. Figure 1.2 shows an example SNP data consisting of 10 sample chromosomes $\{s_1, s_2, ..., s_{10}\}$ and 10 SNPs $\{m_1, m_2, ..., m_{10}\}$, and its corresponding binary matrix.

A real SNP dataset may consist of millions of markers and hundreds of sample chromosomes. In the methods developed in my thesis, various strategies are used to tackle the computation challenges arising in the two problems, i.e., GWA and sample selection. The following sections summarize my contributions.

1.1 Efficient Phylogeny-based Genome-wide Association Mapping Algorithms

Previous studies (Zollner and Pritchard (2005); Mailund et al. (2006); Sevon et al. (2006)) have shown that phylogeny-based association mapping methods outperform single-marker and haplotype-based methods. However, those phylogeny-based methods (Zollner and Pritchard (2005); Mailund et al. (2006); Sevon et al. (2006)) are time-consuming and unable to handle real SNP data. In my thesis, I developed two efficient phylogeny-based association mapping methods, **TreeQA** (Pan et al. (2009)) and **TreeQA+**. Instead of utilizing maximum-parsimony phylogeny or other types of phylogenies (which takes a long time to be inferred from genomic data), TreeQA and TreeQA+ examine the association between the inferred local perfect phylogenetic trees and the phenotype. A perfect phylogenetic tree (Fernandez-Baca (2001)) demonstrates the genetic relationship among a set of haplotypes. For any given set of haplotypes, a unique perfect phylogenetic tree exists if and only if the haplotypes are from a compatible region.

Given a SNP data and a phenotype, both TreeQA and TreeQA+ algorithms take three steps:

1. Identify maximum compatible regions in the SNP data: A compatible region consists of a set of consecutive SNP markers which are all pair-wise compatible by the 4-gamet rule (Hudson and Kaplan (1985)). All genetic variances in a compatible region are introduced by mutations, not from recombination or homoplasy. Each compatible region is maximized on both sides so that it contains as much genetic information as possible.

2. Infer perfect phylogenetic tree for each maximum compatible region: A linear-time algorithm is used to infer local perfect phylogenetic trees for each region (Agarwala

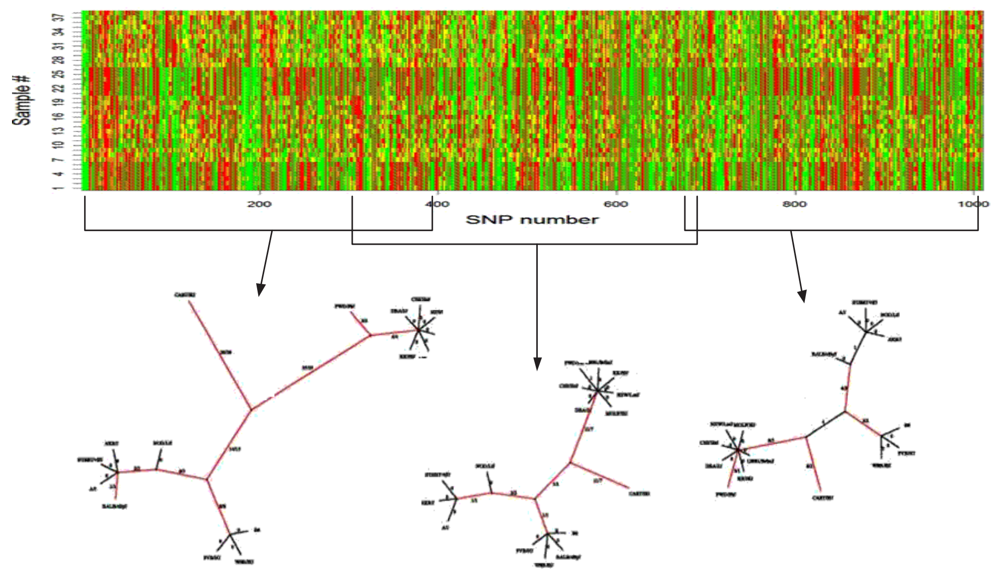


Figure 1.3: Example of maximal compatible regions and perfect phylogenetic trees

et al. (1995)). A piece of SNP data, the identified maximal compatible regions and their corresponding perfect phylogenetic trees are shown in Figure 1.3.

3. Examine the association between each tree and the phenotype: By removing combinations of edges of the tree, both TreeQA and TreeQA+ enumerate all partitions indicated by each tree, but use different statistical models and methods to examine the association between each partition and the phenotype.

In Step 3 of the above framework, genetic relationships among the samples are incorporated into the association analysis by examining all the partitions of samples indicated by the perfect phylogenetic trees. TreeQA and TreeQA+ utilize different models and tests to examine the partitions:

- TreeQA utilizes F-test and permutation test to calculate the significance of the association between each partition and the phenotype. Moreover, TreeQA is able to detect and remove outliers indicated by the tree topology and search for associations in sample subspaces.

- TreeQA+ improves TreeQA by utilizing the Brownian motion and maximum likelihood model to incorporate sample correlations induced by trees. The correlations violate the sample independence assumption and can bias the significance of the association. Ignoring the correlations may cause spurious association to be detected.

Even though both TreeQA and TreeQA+ utilize a linear-time tree inference algorithm, they are still facing the computational challenges caused by test calculation and permutation test. Several efficiency optimizations are developed in the two algorithms.

In particular, for TreeQA:

1. Identical partitions indicated by different trees are stored and retrieved in a prefix tree. The association score of a partition is only calculated once.

2. The intermediate computations in permutation tests are maximally reused by redesigning the calculation.

And for TreeQA+:

1. The number of permutations conducted in each permutation test is varied based on the current best association.

2. The calculation in the maximum likelihood model is optimized to avoid repeated calculations.

In my thesis, extensive experiments are conducted on both synthetic and real SNP data. The results show that TreeQA and TreeQA+ are more robust and effective than single marker and haplotype-based methods, and are more efficient than the previous phylogeny-based methods such as TreeLD (Zollner and Pritchard (2005)).

I further extended TreeQA and developed the TreeNL algorithm which is applied to correlation clustering in high dimensional data. In quantitative data of any domains, the TreeNL method is used to detect sample subspaces where a subset of features are correlated with each other, e.g., a target feature has dependency on other features. This extended application differs from TreeQA in the following three aspects.

1. Hierarchy clustering algorithms are used to generate the tree structures representing sample relationships.

2. All feature subsets are considered in the association test, instead of only considering consecutive features.

3. Every feature is considered as a potential target feature (which may have dependency on other features).

In my thesis, I applied TreeNL on real consensus data and found interesting correlated feature subsets.

1.2 Sample Selection based on Genetic Diversity

Sample selection is closely related to the genome-wide association mapping problem. In my thesis, the maximum-diversity sample selection is formalized as: select a minimum sample subset which can at least retain ρ percent of the genetic diversity. In the studies, the genetic diversity is measured by the sample variation retained on the set of genetic markers. In a biallelic SNP dataset, the genetic diversity is measured as the percentage of SNP markers of which both alleles occur in the selected samples. In non-biallelic genomic data, the diversity is measured by information-theoretic measurements such as mutual information (Guiasu (1977)).

In my thesis, I first developed algorithms to tackle the sample selection problem on biallelic SNP data. In this case, the problem of maximum-diversity sample selection is proved to be NP-complete via reduction from the Set Cover problem (Cormen et al. (2001)). The reduction is sufficiently tight so that greedy approximations to Set Cover directly apply to maximizing diversity. There is a well known greedy algorithm for the

Set Cover problem (Cormen et al. (2001)). It chooses the subset that maximizes the increase in coverage in each step until all the elements are covered. Therefore, I developed the *PGDS* (Pan et al. (2007)) algorithm which takes a similar greedy searching scheme. *PGDS* chooses the sample that maximizes the increase in the diversity retained in each step. The *PGDS* algorithm is different from the general Set Cover algorithm in two aspects:

1. *PGDS* is restarted with each sample and it picks the smallest subset from the n subsets generated, where n is the number of samples. Because a greedy algorithm cannot pick the best first sample based on diversity since every single sample provides zero diversity.

2. The *PGDS* algorithm may stop once the genetic diversity retained in the selected sample subset exceeds the minimum threshold.

The *PGDS* algorithm is efficient on real SNP data as demonstrated in my thesis. However, it does not guarantee the optimal solution. Therefore, I also developed an exhaustive-searching algorithm in my thesis, $K\rho DS$ (Pan et al. (2007)), which is guaranteed to get the optimal solution efficiently in biallelic SNP data. $K\rho DS$ uses *PGDS* as its pre-processing step. With the size of the possible minimum subset, K, reported by *PGDS*, $K\rho DS$ searches all possible combinations of samples up to size K in an enumeration tree. By imposing an order on the samples, the $K\rho DS$ algorithm is able to perform a systematic search by enumerating all combinations, i.e., no combination is missed or revisited. Because of the exhaustive search scheme, $K\rho DS$ guarantees to find the optimal solution. In the worst case, the $K\rho DS$ algorithm has a searching space of size $O(2^n)$ and takes exponential time. In order to accelerate the search, the $K\rho DS$ algorithm uses several pruning strategies to prune the searching space as follows:

- Dynamically Limit the Size of the Minimum Sample Subset: As mentioned above, $K\rho DS$ only searches all sample subsets up to size K which is reported by *PGDS*.

During the search, the value of K is also updated to be the size of the smallest subset found so far (i.e.retains the minimum percentage of diversity). All remaining sample subsets of larger sizes can be pruned from the enumeration tree without further examination.

- Order Samples by Pair-wise Diversity: The runtime of $K\rho DS$ depends on how deep it searches into the enumeration tree. $K\rho DS$ can use less time to find the optimal solution if it can find a qualified subset in the earlier stages of the enumeration. Therefore, $K\rho DS$ orders the samples at each level of the enumeration according to their pair-wise diversity so that it has a larger chance to find a qualified subset in the early stages.

- Estimate a Branch Upper Bound on Diversity: During the enumeration, $K\rho DS$ estimates the maximal diversity that can be found in the sample subsets in the current branch. An easy way to get a maximal diversity of a branch is to calculate the diversity of the sample subset consisting of all the samples which will be enumerated in that branch. Obviously, all the other sample subsets in the branch can only achieve lower diversities than this particular subset. Therefore, if the maximal diversity is less than the threshold ρ, $K\rho DS$ can safely prune the branch.

- Refine the Branch Upper Bound on Diversity: The maximal diversity of a branch is overestimated when it is calculated using all the samples in the branch. With the knowledge of the size of the current minimum subset, K, $K\rho DS$ can refine the upper bound to be the maximal diversity of any K-sample subsets in the current branch.

The experimental results in my thesis show that using these four pruning strategies together can dramatically speed up the $K\rho DS$ algorithm. And the 4^{th} strategy is the most efficient one.

Both $K\rho DS$ and $PGDS$ only work on biallelic SNP data. For non-biallelic data, I developed an information-theoretic sample selection algorithm, REP (Pan et al. (2005a)), to search maximal diversity subsets. Mutual information (Guiasu (1977)) is used to measure the diversity retained in a sample subset. The REP algorithm takes a greedy scheme. In each step, it chooses the sample that can maximize the increase in mutual information until the percentage of mutual information retained exceeds a minimum threshold. Because of the monotonicity property of mutual information of a growing sample subset, the REP algorithm can always find a sample that increases the mutual information in each step. Therefore, the algorithm is guaranteed to terminate in finite steps. Due to the greedy searching scheme, REP does not guarantee the optimal solution. However, experiments in my thesis demonstrate that the sample subsets selected by REP are near optimal.

1.3 Thesis Statement

Efficient and effective algorithms can be developed for the following two analysis tasks on large genomic data.

- Phylogeny-based genome-wide association mapping: I developed the TreeQA and TreeQA+ algorithms which are efficient Phylogeny-based GWA methods. I also extended the idea to general data mining tasks and developed the TreeNL algorithm.

- Maximum-diversity sample selection: I developed efficient algorithms for both biallelic data (the $K\rho DS$ and $PGDS$ algorithms) and non-biallelic data (the REP algorithm).

1.4 Thesis Outline

The thesis is organized as follows:

- The TreeQA algorithm is presented in Chapter 2.
- The TreeQA+ algorithm is presented in Chapter 3.
- The extended TreeQA algorithm, TreeNL, and its application in correlation clustering are discussed in Chapter 4.
- The algorithms for maximum-diversity sample selection on biallelic SNP data are presented in Chapter 5.
- The REP algorithm for maximum-diversity sample selection on non-biallelic data is presented in Chapter 6.
- Chapter 7 concludes my thesis work and outlines the future work.

Chapter 2

TreeQA: Phylogeny-based Genome-wide Association Mapping

2.1 Introduction

Genome wide association (GWA) mapping locates genes or narrows regions in the genome that have significant statistical connections to phenotypes of interest. The discovery of these genes and regions offers the potential to increase understanding of biological processes controlling manifestation of phenotypes.

The most frequent genetic variants are single nucleotide polymorphisms (SNPs), in which a single nucleotide in the genome differs between individuals within a species. With the development of low-cost genotyping technologies, extensive SNP data can be cheaply and efficiently produced, which further increases the computational complexity of GWA mapping. Thus, there is an evident need for fast and effective GWA mapping methods.

Existing methods of association mapping look for similarities among samples (chromosomes, haplotypes, etc.) that are correlated with the phenotypes. If strong associations are present, the variance of the phenotype within groups of similar samples is substantially smaller than the variance over all samples.

For example, in single marker-based (Pe'er et al. (2006); Akey et al. (2001)) and haplotype-based association mapping (Wang and Paigen (2005); Toivonen et al. (2000); Waldron et al. (2005)), samples are grouped according to their genetic variation at a single marker or a set of markers. For case/control phenotypes, markers that can divide samples into (almost) pure classes are reported. Though these methods employ different strategies for grouping samples, the derived groups are evaluated without further consideration of the intergroup similarities or alternate groupings.

In observation of this, tree-based association methods (Mailund et al. (2006); Sevon et al. (2006); Zollner and Pritchard (2005)) utilize phylogenies constructed over the samples. The phylogeny tree is a rich yet compact representation of genetic similarities of the samples. It provides sensible groupings of samples at multiple resolutions. However, the existing methods either handle only case/control phenotypes (Mailund et al. (2006); Sevon et al. (2006)) or do not scale to GWA mapping (Zollner and Pritchard (2005)).

In this chapter, we introduce TreeQA, a tree-based quantitative GWA mapping algorithm. TreeQA utilizes local perfect phylogeny trees constructed in genomic regions exhibiting no evidence of historical recombination by the 4-gamete test (Hudson and Kaplan (1985)). Given a perfect phylogeny, TreeQA evaluates all implied groupings and finds the strongest associations to the phenotype. Furthermore, TreeQA can identify and remove outliers during association analysis.

A brute-force implementation consists of a double loop: for every phylogeny tree, and for every grouping represented by the tree, we conduct a separate ANOVA test to measure its association to the phenotype, and keep track of the best groupings and trees. This approach is inefficient and prone to multiple test errors (Miller (1981)). Both the number of trees and number of groupings per tree can be very large[1]. This large number of possible groupings requires many ANOVA tests, which is not only expensive computationally, but also gives rise to spurious associations[2]. Thus, permutation tests

[1]For example, the number of trees can exceed tens of thousands in a chromosome-wide association study. And there are up to 2^{2n-2} groupings that can be generated from a tree of n samples.

[2]With ε error rate, the risk of reporting at least one spurious association from x tests is $1-(1-\varepsilon)^x$.

are necessary to ensure the statistical significance of the discovered associations, which will further increase the computational burden.

TreeQA exploits the following properties:

1. Groupings generated from the same tree obey a partial order, thus allowing reuse of intermediate computations.

2. A grouping may be derived from different trees, but only need to be evaluated once.

3. Different phenotype permutations may share a substantial number of common computations that need to be computed only once. Thus, TreeQA employs two prefix-tree structures (Cormen et al. (2001)) to organize all observed sample subsets and groupings to facilitate the caching and retrieval of reusable computations and guide the enumeration and evaluation of groupings.

As a result, TreeQA is able to handle quantitative GWA mapping very efficiently and is more effective and robust in association mapping than previous methods.

2.2 Related Work

Single-marker association mapping (Pe'er et al. (2006); Akey et al. (2001)) considers the sample groupings induced independently by each single marker. Statistical tests such as χ^2 and F-tests are used to measure the association between the phenotype and each grouping. These methods are computationally efficient, however, they do not utilize the additional information content carried by haplotypes over single markers.

To address this shortcoming, haplotype-based methods have been developed. HAM (McClurg et al. (2006)) considers combinations of three consecutive SNPs along the genome. QHPM (Onkamo et al. (2002)) uses frequent pattern mining methods to find haplotype patterns in the data, upon which sample groupings are created and evaluated. HapMiner (Wang and Paigen (2005)) clusters samples using consecutive subsets of markers, and then assess the phenotype's association strength.

The utility of local phylogenies in association mapping has been recently explored in TreeLD (Zollner and Pritchard (2005)), Blossoc (Mailund et al. (2006)), and TreeDT (Sevon et al. (2006)). These methods use trees to represent sample similarities. Their approach is to exhaustively examine all possible groupings implied by the given phylogenies without explicitly excluding any outliers. Both Blossoc and TreeDT assume simple categorical (binary) phenotypes. TreeLD handles quantitative phenotypes but is not scalable to GWA analysis.

Some other work (Larribea et al. (2002); Morris et al. (2002); Minichiello and Durbin (2006)) uses a global phylogeny structure, e.g., ancestral recombination graph, over all markers in association mapping. However, because of the high computational cost of global phylogeny construction, these methods are not scalable to genome-wide analysis.

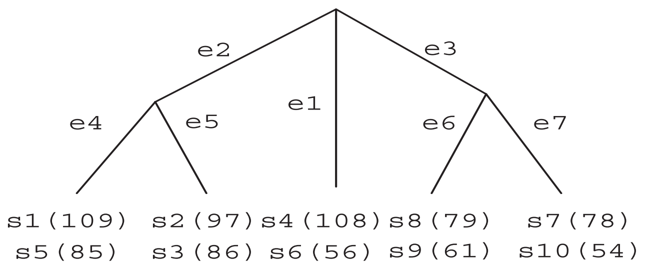
2.3 Preliminaries

We use a binary matrix $H = S \times M$ to represent a SNP dataset, where $S = \{s_1, s_2, ..., s_n\}$ is the set of samples, and $M = \{m_1, m_2, ..., m_z\}$ is the SNP marker set. Each sample is represented by a binary vector, in which '0' represents the majority alleles and '1' represents the minority alleles. We use $f(s_i)$ to denote the phenotype value of a sample s_i and $F(S')$ to denote the phenotype values of samples in a subset S'. An example matrix H containing 10 samples and 10 SNP markers with phenotype is shown in Fig. 2.1(a).

Definition 2.3.1. Compatible region: A consecutive region of the genome is called a compatible region *iff* any pair of markers in that region are compatible by the 4-gamete test (Hudson and Kaplan (1985)). That is, among the 4 possible haplotypes formed by

	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	f(s)
s1	1	0	1	0	0	0	0	0	0	0	109
s2	1	0	0	0	1	0	0	0	1	0	97
s3	1	0	0	0	1	0	0	0	0	0	86
s4	0	0	0	0	0	0	0	0	0	0	108
s5	1	0	1	0	0	0	0	0	0	0	85
s6	0	0	0	0	0	0	0	0	0	0	56
s7	0	0	0	0	0	1	0	1	0	0	78
s8	0	1	0	1	0	1	1	0	1	1	79
s9	0	1	0	1	0	1	1	0	1	0	61
s10	0	0	0	0	0	1	0	1	0	0	54

(a) SNP data & phenotype



(b) Tree $T_{1,8}$

Figure 2.1: Example: a SNP dataset and a perfect phylogeny tree

the two markers, at most three of them occur.

A compatible region is a genomic region exhibiting no evidence of historical recombination. In Fig. 2.1(a), the region from markers m_1 to m_8 is a compatible region. We use $C_{u,v}$ to denote a compatible region from markers m_u to m_v.

Definition 2.3.2. Maximal Compatible region: A compatible region is a maximal compatible region *iff* it can not be extended on either side to include more SNPs and remains compatible.

Definition 2.3.3. Perfect Phylogeny Tree: A phylogeny tree for a set of samples is *perfect* if the phylogeny avoids homoplasy. Every SNP is introduced by a mutation and is represented by an edge of the tree. Given a genomic region, a perfect phylogeny exists *iff* the region is a compatible region.

We use $T_{u,v}$ to denote the perfect phylogeny tree of compatible region $C_{u,v}$. Given

$C_{1,8}$ in Fig. 2.1(a), its tree $T_{1,8}$ is shown in Fig. 2.1(b). All samples are at the leaf nodes. Samples having identical haplotypes in the region share the same leaf node in the tree, e.g., s_1 and s_5. Each internal node represents a hypothetical common ancestor of a subset of samples. Each edge uniquely corresponds to a SNP (or a historical mutation). Interested readers may refer to paper (Agarwala et al. (1995)) for inferring perfect phylogenies from a set of SNPs.

Let $E(T_{u,v}) = \{e_1, e_2, ..., e_p\}$ denote the set of edges in $T_{u,v}$. The removal of each edge partitions the samples into two subsets denoted by $S^{(0)}(e_i)$ and $S^{(1)}(e_i)$. Given a tree $T_{u,v}$, we can generate $2|E(T_{u,v})|$ sample subsets by removing each edge separately. We denote this set of sample subsets by $S^{(E)}(T_{u,v})$, $S^{(E)}(T_{u,v}) = \{S^{(j)}(e_i)|j = \{0, 1\}, e_i \in E(T_{u,v})\}$.

Definition 2.3.4. A grouping of a sample subset S', $G(S')$, is formed by a set of disjoint subsets of S', $G(S') = \{S'_1, S'_2, ..., S'_k\}, S'_i \subset S', S'_i \cap S'_j = \emptyset, \bigcup_{i=1}^{k} S'_i = S'$. Given a tree $T_{u,v}$, we say a grouping $G(S')$ *follows* $T_{u,v}$ *iff* $\forall S'_i \in G(S')$, $S'_i \in S^{(E)}(T_{u,v})$.

For example, grouping $G(S') = \{\{s_1, s_5, s_2, s_3\}, \{s_8, s_9, s_7, s_{10}\}\}$ follows the tree in Fig. 2.1(b), while grouping $G(S') = \{\{s_1, s_2\}, \{s_8, s_4\}\}$ does not.

Definition 2.3.5. Given a sample subset S', $G_1(S')$ is called a *parent-grouping* of $G_2(S')$ ($G_2(S')$ called a *child-grouping* of $G_1(S')$) *iff* $\forall S'_i \in G_1(S')$

$$\exists S'_j \in G_2(S'), s.t. S'_i = S'_j.\ OR\ \exists\{S'_{j_q}|S'_{j_q} \in G_2(S'), q = 1, ..., u\}, s.t. S'_i = \bigcup_{q=1}^{u} S'_{j_q}$$

A child-grouping represents a finer partition of its parent-grouping on the same set of samples. For example, grouping $\{\{s_1, s_5, s_2, s_3\}, \{s_4, s_6\}\}$ is the parent-grouping of $\{\{s_1, s_5\}, \{s_2, s_3\}, \{s_4, s_6\}\}$. We summarize the notations in Table 4.1.

Association between a Compatible Region and a Phenotype

We use the one-way ANOVA test with permutations to measure the association between a grouping of samples and a quantitative phenotype. To accelerate the execution, we

Table 2.1: Summary of Notations

S, s_i, S'_i	the sample set, a sample, a subset of samples
M, m_i	the marker set, a marker
H	a binary matrix representing the data
$C_{u,v}$	a compatible interval of H
$f(s_i)$	phenotype value of sample s_i
$F(S'_i)$	the set of phenotype values of the samples in S'_i
$G_i(S')$	a grouping of a sample subsets S'
$T_{u,v}$	the perfect phylogeny tree of $C_{u,v}$
$E(T_{u,v})$	the edge set of $T_{u,v}$
$T'_{u,v}(e_i, s_j)$	the subtree rooted at node s_j after removing edge e_i
$S^{(E)}(T_{u,v})$	the set of sample subsets implied in tree $T_{u,v}$ (leaf-sets)

re-derive the formula of the ANOVA test.

Given a grouping $G(S') = \{S'_1, ..., S'_k\}$, for every $S'_i \in G(S')$, we calculate

$$SQ(S'_i) = \sum_{s_j \in S'_i} f(s_j)^2, \quad SM(S'_i) = \sum_{s_j \in S'_i} f(s_j) \tag{2.1}$$

$$SSE_i = SQ(S'_i) - SM(S'_i)^2/|S'_i|, \quad SSB_i = SM(S'_i)^2/|S'_i| \tag{2.2}$$

Combining all subsets together, we have $MM = \frac{1}{|S'|}\sum_{i=1}^{k} SM(S'_i)$ and

$$MSE = \frac{1}{|S'| - k}\sum_{i=1}^{k} SSE_i, \quad MSB = \frac{1}{k-1}(\sum_{i=1}^{k} SSB_i - |S'| \cdot MM^2) \tag{2.3}$$

We obtain a base score for grouping $G(S')$

$$\digamma_0(G(S')) = \frac{MSB}{MSE} \tag{2.4}$$

A higher score indicates a stronger association between the grouping and the phenotype. Given the tree and the data in Fig. 2.1 and the following two groupings: $G(S'_1) = \{\{s_2, s_3\}, \{s_4, s_6\}, \{s_8, s_9\}\}$, $G(S'_2) = \{\{s_2, s_3\}, \{s_8, s_9\}\}$, the scores are $\digamma_0(G(S'_1)) = 0.44$, $\digamma_0(G(S'_2)) = 4.16$. Thus, grouping $G(S'_2)$ has a stronger association with the phenotype than grouping $G(S_1')$.

To correct the multiple test errors, we apply a permutation test on $G(S')$ to calculate a significance score. To permute the phenotype, the phenotype values in $F(S')$ are randomly re-assigned to samples in S'. Then we calculate an $\digamma$-score using the permuted phenotype following Eqs. 5.1 to 3.13.

Assume that we conduct $nPerm$ random permutations in total, for each permutation, we get score $\digamma_j (j = 1...nPerm)$. Among the $nPerm$ $\digamma$-scores, let p be the number of scores which are greater than or equal to the base score $\digamma_0(G(S'))$, i.e., $p = |\{\digamma_j | \digamma_j \geq \digamma_0(G(S')), j \in 1...nPerm\}|$. Then the significant score (P score) of $G(S')$ is

$$P(G(S')) = \log_{10}\left(\frac{nPerm}{p}\right) \tag{2.5}$$

A higher P score indicates that the association between grouping $G(S')$ and the phenotype is more significant.

Definition 2.3.6. The association between a compatible region and a phenotype: For a compatible region $C_{u,v}$, the highest P score achieved by any grouping following $T_{u,v}$ is regarded as the P score of $C_{u,v}$. The P score represents the association between the compatible region and the phenotype,

$$P(C_{u,v}) = max\{P(G_j(S'))|\forall G_j(S')\ follows\ T_{u,v}, S' \subseteq S\}. \tag{2.6}$$

Problem Definition: Given a SNP data and a quantitative phenotype, calculate the P-score of every maximal compatible region and report the most significant ones.

2.4 TreeQA Algorithm

TreeQA takes two major steps:

1. Identify maximal compatible regions in the genome and construct the perfect phy-

logenies of the regions.

2. Compute the association between each compatible region and the phenotype.

2.4.1 Maximal Compatible Region and Phylogeny Construction

TreeQA scans the SNP markers in a left to right order. In order to find the maximal compatible regions, it continuously extends the current region by adding the next marker until the new marker is incompatible with some markers in the region. And it maximizes the overlap between two consecutive regions. Assume that the current compatible region is $C_{u,v}$, and marker m_{v+1} is incompatible with markers $m_{i_1}, ..., m_{i_k}$, $u \leq i_1 < ... < i_k \leq v$, then TreeQA starts the next compatible region at marker m_{i_k+1}. For each maximal compatible region, TreeQA utilizes the inferring algorithm Agarwala et al. (1995) to construct a local perfect phylogeny tree. Both procedures have linear time complexity with respect to the number of markers and the number of samples.

2.4.2 Association Computing

In the second step, TreeQA takes as input a quantitative phenotype and a set of local perfect phylogenies. It considers all possible groupings following the phylogenies and systematically explores the search space of these groupings in a carefully designed order such that intermediate computations can be maximally reused.

According to Definition 2.3.4, any grouping of a sample subset[3] that follows a tree $T_{u,v}$ can be created from non-overlapping subsets in $S^{(E)}(T_{u,v})$. By utilizing the lexicographical order[4] of subsets in $S^{(E)}(T_{u,v})$, TreeQA can enumerate and evaluate all combinations of non-overlapping subsets systematically.

[3]Considering groupings of a sample subset allows TreeQA to exclude potential outliers from the ANOVA test.

[4]Any other ways of defining a total order of the subsets would also work.

TreeQA enumerates all groupings via a depth-first recursive procedure. TreeQA extends the current grouping by including a new sample subset which does not overlap with any subsets in the current grouping. The association of each new grouping to the phenotype via a permutation test is computed. The P score of the corresponding maximal compatible region is updated accordingly. The enumeration continues recursively for each newly extended grouping.

Consider the tree in Figure 2.1. There are 14 sample subsets in $S^{(E)}(T_{1,8})$. Assume that the subsets have the following order,

$se_1 = \{s_1, s_5\}, se_2 = S - se_1,\ se_3 = \{s_2, s_3\}, se_4 = S - se_3, se_5 = \{s_4, s_6\}$

$se_6 = S - se_5,\ se_7 = \{s_8, s_9\}, se_8 = S - se_7, se_9 = \{s_7, s_{10}\}, se_{10} = S - se_9$

$se_{11} = \{s_1, s_5, s_2, s_3\}, se_{12} = S - se_{11}, se_{13} = \{s_8, s_9, s_7, s_{10}\}, se_{14} = S - se_{13}$

TreeQA first generates a grouping containing se_1 only. Among the remaining sample subsets, $\{se_2, se_3, se_5, se_7, se_9, se_{12}, se_{13}\}$ do not overlap with se_1. In the next step, a grouping $\{se_1, se_2\}$ is formed by adding se_2 into the current grouping and its P score is calculated. $P(C_{1,8})$ is updated accordingly. Since all other sample subsets overlap with se_1 or se_2. Thus, no new grouping can be extended from $\{se_1, se_2\}$. Then, TreeQA examines the next grouping extended from $\{se_1\}$, $\{se_1, se_3\}$, and all groupings extended from it. After examining all groupings containing se_1, TreeQA will start from the grouping $\{se_2\}$ and extend it recursively to generate all groupings containing se_2 but not se_1. This process continues until all distinct groupings are enumerated.

The pseudocode code of TreeQA is in Fig. 2.2.

2.4.3 Effective Permutation

We found that more than 90% of the execution time of TreeQA is spent in permutation tests. Given a grouping $G(S')$, a permutation test is conducted in two steps: 1) ran-

Main Routine
Input:

- Sample set S.
- A list of local perfect phylogenies, $\{T_{u,v}\}$.
- $F(S)$, quantitative phenotype values.
- *nPerm*, number of permutations.

Output: $P(C_{u,v})$ of all compatible regions.
Method:

```
(1)  for each $T_{u,v}$ in the set of trees
(2)      $SE = \{se_i | se_i \in S^{(E)}(T_{u,v})\}$
(3)      $P(C_{u,v}) = 0$.
(4)      for j=1 to $|SE|$
(5)          add $se_j$ to curlist.
(6)          add $se_l$ to remlist if $l > j, se_j \cap se_l = \emptyset$
(7)          Enumerate(curlist,remlist,nPerm,$P(C_{u,v})$)
(8)      report $P(C_{u,v})$.
```

Subroutine: Enumerate(*curlist*,*remlist*,*nPerm*,$P(C_{u,v})$)
Method:

```
(1)  for each remaining subset $se_l$ in remlist
(2)      create grouping $G(S')$=curlist$\cup\{se_l\}$.
             $S' = \bigcup se_q, se_q \in curlist \cup \{se_l\}$
(3)      calculate base score $F_0(G(S'))$.
(4)      $p = 0$.
(5)      for j=1 to nPerm
(6)          random permutation and re-calculate $F$.
(7)          if($F \geq F_0(G(S'))$), $p = p + 1$.
(8)      $p = log_{10}(\frac{nPerm}{p})$.
(9)      $P(C_{u,v}) = max(p, C_{u,v})$.
(10)     remlist=remlist-$\{se_l\}$
(11)     curlist'=curlist$\cup\{se_l\}$. remlist'=remlist.
(12)     remove subsets in remlist' that overlap with $se_l$.
(13)     Enumerate(curlist',remlist',nPerm,$P(C_{u,v})$)
```

Figure 2.2: The TreeQA Algorithm

domly re-assigning the phenotype values in $F(S')$ to samples in S'; 2) calculating the corresponding $\mathcal{F}$ score by Eq. 3.13.

Given a subset S', both steps take $O(|S'|)$ time. TreeQA exploits maximal reusability of intermediate computation shared by permutation through the following two optimizations:

1) ***inTree***: Common computation units shared by permutation tests of parent/child-groupings in a tree.

2) ***amgTree***: Common computation units shared by permutation tests on groupings following multiple trees.

We use two global prefix-tree structures (Cormen et al. (2001)), $Tree_{grouping}$ and $Tree_{subset}$ to organize groupings and sample subsets examined thus far respectively to enable effective permutation tests.

inTree: Effective permutation tests within a tree

A pair of parent/child-groupings always involve the same set of samples. Let S' denote a set of samples. For the permutation tests of the parent/child groupings of S', instead of re-assigning the phenotype values in $F(S')$ independently for each grouping, they can share the same set of random permutations of $F(S')$.

For example, given the example in Fig. 2.1 and a pair of parent/child-groupings, $G_1(S') = \{\{s_1, s_5, s_2, s_3\}, \{s_8, s_9, s_7, s_{10}\}\}$ and $G_2(S') = \{\{s_1, s_5\}, \{s_2, s_3\}, \{s_8, s_9, s_7, s_{10}\}\}$, their $\mathcal{F}_0$ scores are: $\mathcal{F}_0(G_1(S')) = 9.79$ and $\mathcal{F}_0(G_2(S')) = 4.32$. Assume that after a random permutation, the new phenotype values for the samples are: $f(s_1) = 85$, $f(s_2) = 79$, $f(s_3) = 109$, $f(s_5) = 61$, $f(s_7) = 86$, $f(s_8) = 97$, $f(s_9) = 78$, $f(s_{10}) = 54$. Using this new assignment, we can calculate the new $\mathcal{F}$ scores for both groupings: $\mathcal{F}(G_1(S')) = 0.12$ and $\mathcal{F}(G_2(S')) = 0.7$. By reusing the phenotype permutation between $G_1(S')$ and $G_2(S')$, we save $O(|S'|)$ runtime in each permutation.

A child-grouping represents a finer partition of sample subsets in its parent-grouping.

We say a grouping is at the finest level if it does not have any child-groupings. For example, given the tree in Fig. 2.1(b) and the two groupings $G_1(S')$ and $G_2(S')$ used above, grouping

$$G_3(S') = \{\{s_1, s_5\}, \{s_2, s_3\}, \{s_8, s_9\}, \{s_7, s_{10}\}\}$$

is the child-grouping of both $G_1(S')$ and $G_2(S')$ and is at the finest level while $G_2(S')$ is a finer partition of $G_1(S')$.

We use a global prefix-tree $Tree_{grouping}$ to index all groupings and maintain the parent/child relationship through auxiliary links (from a child-grouping to its parent-groupings). For each permutation of the phenotype, the F scores of a finest grouping and all of its parent-groupings are calculated together. We examine the finest grouping immediately followed by the examination of its parent groupings for maximum computation reuse. If a finest child-grouping has n parent-groupings, we save $O(n|S'|)$ time in each permutation.

Given a tree T_i, each grouping $G_j(S')$ is inserted into or retrieved from $Tree_{grouping}$ after Step 2 in **Enumerate**(). If $G_j(S')$ exists in $Tree_{grouping}$ and the P score is already calculated which means $G_j(S')$ has been examined, *amgTree* is used and the subroutine jumps to Step 9 (discussion about *amgTree* is in Section 2.4.3). If grouping $G_j(S')$ is new, we insert $G_j(S')$ into $Tree_{grouping}$ and generate its child-groupings recursively till the finest level, say $G_l(S')$.

If $G_l(S')$ exists in $Tree_{grouping}$ (that is, it was examined before), we insert the new leaf node of $G_j(S')$ to the linked list headed by the leaf node of $G_l(S')$. If $G_l(S')$ is not in $Tree_{grouping}$ (that is, this is the first time it is examined), we insert $G_l(S')$ into the tree first, create a linked list headed by the new leaf node of $G_l(S')$, and insert the leaf node of $G_j(S')$ in the linked list. Since a parent-grouping may be inserted into tree $Tree_{grouping}$ before its child-groupings and a child-grouping could have multiple parent-groupings, the calculation of P score (from Steps 3 to 8) are deferred until all groupings in T_i are enumerated and inserted into $Tree_{grouping}$.

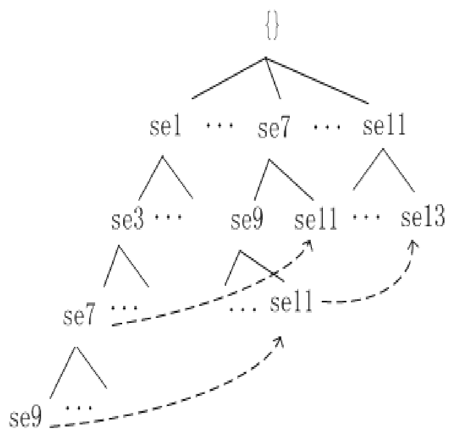


Figure 2.3: A fragment of the $Tree_{grouping}$ tree after enumerating the tree in Figure 2.1

For example, given the tree in Figure 2.1, after enumerating all groupings, a fragment of $Tree_{grouping}$ is shown in Figure 2.3. Parent/child-groupings are in linked lists represented by dotted arrows, of which the finest groupings (such as $\{se_1, se_3, se_7\}$) are at the head.

amgTree: Effective permutation among trees

The same grouping occurs repeatedly in different trees. We only need to compute its P score at its first occurrence. We use $Tree_{grouping}$ to store and retrieve the P score of all examined groupings. If the grouping formed by TreeQA can be found in $Tree_{grouping}$, its P score is directly used. Otherwise, its P score is calculated and stored in $Tree_{grouping}$.

Based on our experiments on real data, using *amgTree* alone can reduce 40%−50% of the execution time. When using *inTree* and *amgTree* together, we can reduce 70%−80% of the execution time.

2.4.4 Reuse of Intermediate Computation of Statistical Tests

For any sample subset S', $SQ(S')$ and $SM(S')$ calculated using the original phenotype values (with no permutation) may be reused in any grouping containing S' and all its parent-groupings. We denote them by $SQ_0(S')$ and $SM_0(S')$ respectively in the following discussion.

We employ a global prefix-tree $Tree_{subset}$ to keep track of all sample subsets in any groupings examined thus far. Three values are stored at the leaf node corresponding to the subset S': (subset ID, $SQ_0(S')$, $SM_0(S')$).

For example, given the 10 samples and their phenotype values in Fig. 2.1(a), we calculate the base score $\digamma_0$ of grouping $G_1(S') = \{\{s_1, s_5\}, \{s_2, s_3\}, \{s_7, s_{10}\}\}$.

$$SQ_0(S'_{1_1}) = 19106, SQ_0(S'_{1_2}) = 16805, SQ_0(S'_{1_3}) = 9000.$$

$$SM_0(S'_{1_1}) = 194, SM_0(S'_{1_2}) = 183, SM_0(S'_{1_3}) = 132.$$

$$\digamma_0(G_1(S')) = 547.17/212.17 = 2.58.$$

The SQ_0 and SM_0 values of the three subsets are then stored in $Tree_{subset}$. Given a parent-grouping of $G_1(S')$, $G_2(S') = \{\{s_1, s_5, s_2, s_3\}, \{s_7, s_{10}\}\}$, we can retrieve the values of SQ_0 and SM_0 and use them to calculate $\digamma_0(G_2(S'))$,

$$SQ_0(S'_{2_1}) = SQ_0(S'_{1_1}) + SQ_0(S'_{1_2}) = 35911, SQ_0(S'_{2_2}) = SQ_0(S'_{1_3}).$$

$$SM_0(S'_{2_1}) = SM_0(S'_{1_1}) + SM_0(S'_{1_2}) = 377, SM_0(S'_{2_2}) = SM_0(S'_{1_3}).$$

$$\digamma_0(G_2(S')) = 1064.08/166.69 = 6.38.$$

The reuse of $SQ_0(S')$ and $SM_0(S')$ between parent/child groupings may work in conjunction with the *inTree* effective permutation. Besides, $SQ_0(S')$ and $SM_0(S')$ can also be reused by any groupings that contain the subset S'.

2.5 Experimental Results

We compare TreeQA with the following algorithms:

1. **SMA**, our implementation of the Single Marker Association algorithm (Pe'er et al. (2006); Akey et al. (2001)).

2. **HAM**, our implementation of the Haplotype Association Mapping algorithm (McClurg et al. (2006)) that slides a 3-SNP window through the genome

3. **HapMiner** (Wang and Paigen (2005)), downloaded from the website[5].

4. **TreeLD** (Zollner and Pritchard (2005)), downloaded from the website[6].

Both **SMA** and **HAM** use the one-way ANOVA test for fair comparison.

QHPM (Onkamo et al. (2002)) is not used for comparison because it is not scalable to large data sets. Blossoc (Mailund et al. (2006)) and TreeDT (Sevon et al. (2006)) are not used because they require categorical phenotypes.

2.5.1 Experiments on Simulated Data

We use Coasim (Mailund et al. (2005)) to simulate 1000 sequences with scaled recombination rate $\rho = 400$ that corresponds roughly to 10 cM. 10,000 SNP markers are placed uniformly at random over the sequences.

SNP markers on the sequences are randomly selected as causative loci with one, two and three causative mutations. The first SNP is always selected randomly from all SNPs. In the cases of two and three mutations, the second and third causative SNPs are selected from compatible SNPs that are located less than 10 SNPs away from the first SNP. Phenotype values are sampled from four Gaussian distributions: $N_1(140, 35)$,

[5]http://vorlon.case.edu/ jxl175/HapMiner.html

[6]http://pritch.bsd.uchicago.edu/treeld.html

$N_2(90, 35)$, $N_3(50, 40)$, and $N_4(10, 35)$. The one-mutation case uses N_1 and N_3. The two-mutation case uses N_1, N_2 and N_3. The three-mutation case uses all four Gaussian distributions. After assigning the phenotype values, all causative SNPs are removed from the data and we randomly select 100 sequences for our experiments.

SMA, HAM and HapMiner output the top one scoring locus as a point estimation of the causative locus, while TreeQA outputs the top one compatible region. We compare the effectiveness of the algorithms by measuring the distance (in cM) from the top one scoring locus or the center of the top one region to the causative SNP (or the average distance to every causative SNP). We call the distance the **Prediction error**.

Since HapMiner can not finish processing 10,000 SNP markers in a reasonable time, we only use the first 1,000 markers of each sequence when applying HapMiner on the simulated data.

The comparison of SMA, HAM, HapMiner and TreeQA is shown in Figure 3.5. The x-axis represents the prediction error (distance) to the causative locus and the y-axis represents the percentage of causative loci which are found in distance less than x. In all three cases, the estimated loci by TreeQA are closer to the causative loci than those by SMA, HAM and HapMiner.

The TreeLD algorithm uses local phylogenies and analyzes quantitative phenotypes. However, TreeLD can only process a very small amount of data in reasonable time. Therefore, we select 36 samples and 20 SNP markers from the simulated data for performance comparison. A one-mutation causative locus is selected from the 20 SNPs. For TreeQA, instead of generating maximal compatible regions as discussed in Sec. 3.2, a compatible region is generated around each SNP and contains up to five SNPs. TreeLD takes about two hours to analyze this small data while TreeQA finishes in seconds. Figure 2.5 plots the results from TreeLD and TreeQA. The x-axis represents the simulated positions in the genome and the y-axis represents the scores of the SNPs. The vertical line demonstrates the causative locus. Both methods detects a peak near the causative

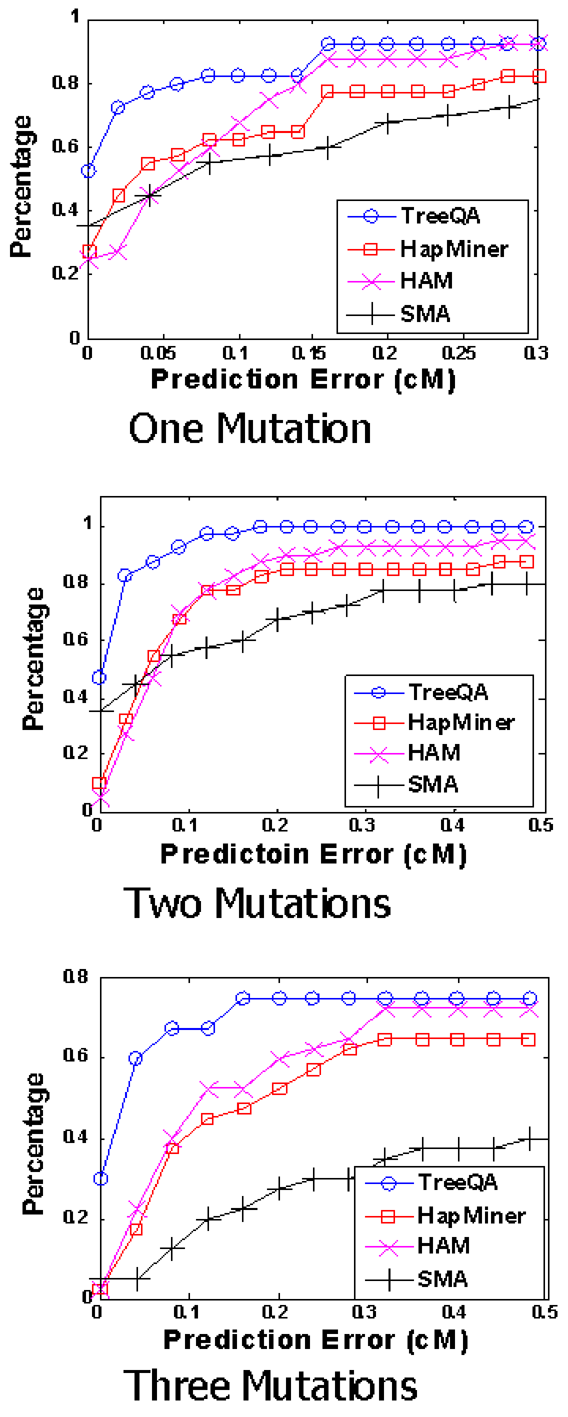


Figure 2.4: Comparison of SMA, HAM, HapMiner and TreeQA on the simulated data

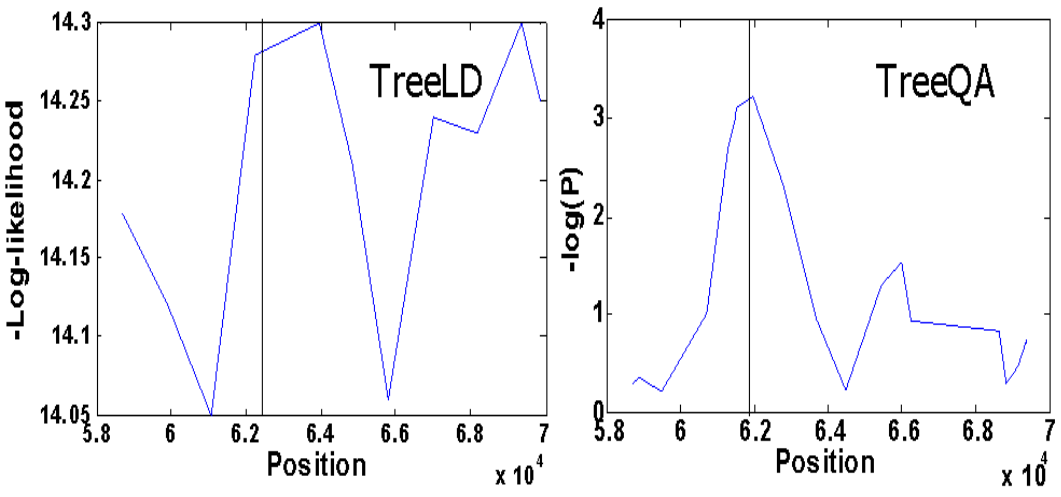


Figure 2.5: Comparison of TreeLD and TreeQA on the simulated data

locus while TreeLD identifies one spurious peak.

2.5.2 Experiments on Mouse Genotype Data

We used a set of mouse genotypes that combines experimental and imputed data[7] (Szatkiewicz et al. (2008)) from the Jackson Laboratory, consisting of 74 samples. The dataset contains over 7 million SNP markers distributed over all 20 chromosomes. We removed wild derived mouse inbred strains since they are quantitatively and qualitatively different than other laboratory inbred strains and we only used in our experiments the remaining 55 samples that have a share set of common ancestral relationships (Yang et al. (2007)).

We used high density lipoprotein cholesterol (HDL-C) levels in blood as the test phenotype, downloaded from the Mouse Phenome Database[8]. Several HDL-C datasets are available, each of which was collected under different conditions, and are thus treated as separate phenotypes. Some candidate genes that may play a role in regulating HDL-C levels are reported in (Wang and Paigen (2002)).

[7]http://cgd.jax.org/ImputedSNPData/imputedSNPs.htm

[8]http://phenome.jax.org/pub-cgi/phenome/mpdcgi?rtn=meas/catlister/req=Cblood+lipids

We apply SMA, HAM and TreeQA on the data and examine how close they can identify the top peak near the locus of those candidate genes.

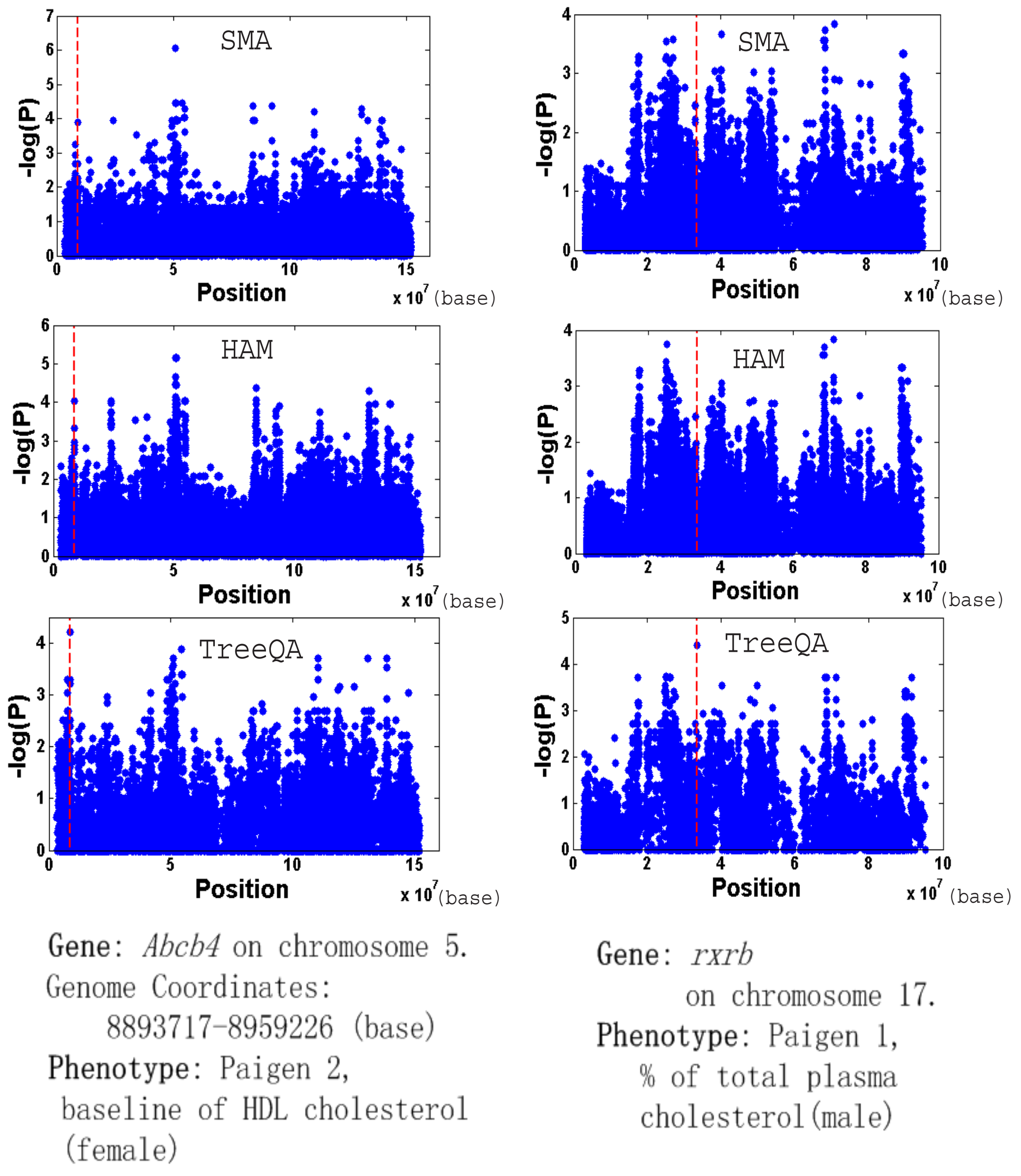


Figure 2.6: Compare SMA, HAM and TreeQA on the mouse genotype data

TreeQA detects top peaks near the locations for over 10 of the candidate genesWang and Paigen (2002), including *Ppara*, *Abcb4* and *Rxrb*. The top peaks reported by SMA and HAM are often far from the locations of these genes. Two of the results are shown in Figure 2.6.

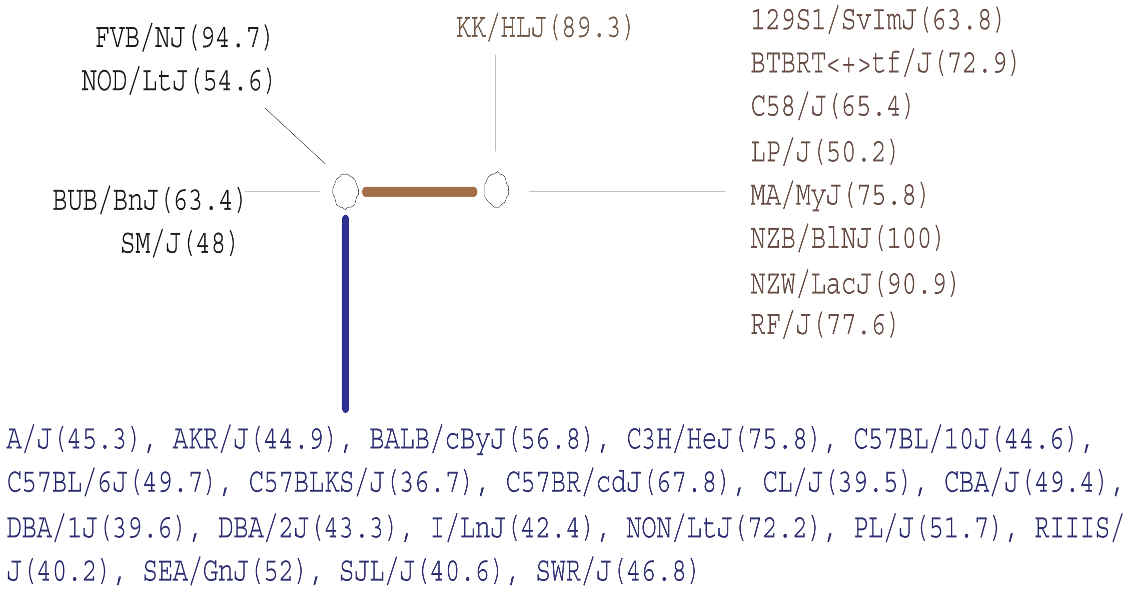

FVB/NJ(94.7)
NOD/LtJ(54.6)
KK/HLJ(89.3)
129S1/SvImJ(63.8)
BTBRT<+>tf/J(72.9)
C58/J(65.4)
LP/J(50.2)
MA/MyJ(75.8)
NZB/BlNJ(100)
NZW/LacJ(90.9)
RF/J(77.6)
BUB/BnJ(63.4)
SM/J(48)
A/J(45.3), AKR/J(44.9), BALB/cByJ(56.8), C3H/HeJ(75.8), C57BL/10J(44.6), C57BL/6J(49.7), C57BLKS/J(36.7), C57BR/cdJ(67.8), CL/J(39.5), CBA/J(49.4), DBA/1J(39.6), DBA/2J(43.3), I/LnJ(42.4), NON/LtJ(72.2), PL/J(51.7), RIIIS/J(40.2), SEA/GnJ(52), SJL/J(40.6), SWR/J(46.8)

Figure 2.7: The perfect phylogeny at the peak point found by TreeQA in Figure 2.6

The perfect phylogeny corresponding to the peak point (compatible region from 8799298 to 8801558 (base)) found by TreeQA around *Abcb4* in Figure 2.6 is plotted in Fig. 2.7. The phenotype values of the samples are in parentheses. Samples with unknown phenotype values are omitted from the tree. The subtree on the right contains samples having high phenotype values while the subtree at the bottom contains samples having low values. Other subtrees are considered as outliers and are excluded from the grouping. SMA and HAM fail to identify the locus because they only examine sample groupings that can be generated from single SNPs or 3-SNP windows, which are a small subset of the groupings examined by TreeQA.

TreeQA takes about 10 minutes to analyze each chromosome which contains around 40000 SNPs on average. SMA and HAM take slightly less time than TreeQA. Both HapMiner and TreeLD are unable to finish in reasonable time.

2.6 Conclusion

In chapter, we present a tree-based quantitative GWA mapping algorithm, TreeQA. TreeQA utilizes local perfect phylogenies in detecting associations. Perfect phylogenies provide sensible groupings of samples at multiple resolutions. TreeQA explores the space of all possible groupings implied by the perfect phylogenies in a carefully designed order so that intermediate computations can be maximally reused. Our experimental results on both simulated and real data show that TreeQA can efficiently conduct quantitative GWA analysis and is more effective than the previous methods.

Chapter 3

TreeQA+: Improving the Power of Phylogeny-based Genome-wide Association Mapping

3.1 Introduction

In Chapter 2, I discussed the TreeQA algorithm which is a phylogeny-based genome-wide association mapping method. The experimental results on both synthetic and real data demonstrates that TreeQA outperforms single marker-based and haplotype-based methods. TreeQA also outperforms the previous phylogeny-based methods such as TreeLD, Blossoc and TreeDT (Zollner and Pritchard (2005); Mailund et al. (2006); Sevon et al. (2006); Larribea et al. (2002); Morris et al. (2002); Minichiello and Durbin (2006)) in terms of runtime and the ability to handle quantitative traits.

However, both TreeQA and other phylogeny-based methods do not consider the sample correlations implied by the tree topologies in the analysis. Ignoring sample correlations can bias the significance of the associations and lead to spurious signals.

For example, three phylogeny trees are plotted in Fig. 4.1. At the leaf nodes, we use "$s_1, s_2, ...$" to represent the samples. The phenotype values are shown in the parentheses. Let's consider the partition created by removing the edge in the middle. We

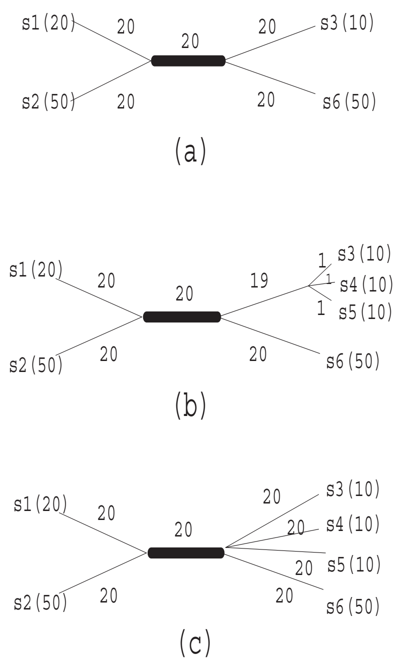


Figure 3.1: The sample correlations affect the significance of the association.

get partitions, $\{\{s_1, s_2\}, \{s_3, s_6\}\}$, $\{\{s_1, s_2\}, \{s_3, s_4, s_5, s_6\}\}$ and $\{\{s_1, s_2\}, \{s_3, s_4, s_5, s_6\}\}$ from the three trees. The mean phenotype values of the left group and right group in these partitions are, $\{35, 30\}$, $\{35, 20\}$ and $\{35, 20\}$. If all samples are assumed to be independent as in the previous methods, the associations between the partitions and the phenotype would be equally strong in trees (b) and (c), and weak in tree (a). However, since s_3, s_4 and s_5 are far more closely related to each other than to the remaining samples (indicated by the short branches between them) in tree (b), it is erroneous to treat them as independent samples in tree (b). In fact, the associations between the partitions and the phenotype should be similarly weak in trees (a) and (b), and relatively strong in tree (c).

Therefore, it is critical to take into account the sample correlations implied by the topology properly in association study. However, this is not a trivial task, especially when we assess the association of the partitions such as $\{\{s_1, s_2\}$,$\{s_3, s_4, s_5\}, \{s_6\}\}$ (created by removing multiple edges).

In this chapter, we introduce TreeQA$^+$, a quantitative GWA mapping algorithm which incorporates the sample correlations modelled by local perfect phylogeny trees. As a phylogeny-based method, TreeQA$^+$ inherits all advantages of TreeQA by examining all groupings induced by a perfect phylogeny (constructed in genomic regions exhibiting no evidence of historical recombination by the 4-gamete test(Hudson and Kaplan (1985))).

In addition, TreeQA$^+$ is more effective and robust than TreeQA by incorporating sample correlations. TreeQA$^+$ adopts the model of Brownian motion (Nelson (1967)) which was previously used to study phylogeny (Edwards and Cavalli-Sforza (1964); Felsenstein (1981)): for any two nodes (samples or hypothetical ancestors) in the phylogeny, if there is no causative mutation happened during the evolution from one node to the other, the difference between the phenotype values of the two nodes should follow a normal distribution with mean 0 and variance proportional to the sum of edge lengthes between them. Thus, any significant deviation from this estimation suggests the existence of some causative mutations during the evolution.

In TreeQA$^+$, a grouping also consists of several non-overlapping subtrees created by removing edges from a perfect phylogeny tree. TreeQA$^+$ utilizes Felsenstein's tree pruning method (Felsenstein (1981)) to estimate the phenotype values of hypothetical ancestors (intermediate nodes) in each subtree. Then the estimated phenotype values at the two adjacent nodes of each removed edge are examined under the assumption of Brownian motion. A significant deviation between the two nodes implies a strong association between the grouping and the phenotype. For each phylogeny, TreeQA$^+$ finds the strongest association between its induced groupings and the phenotype.

A brute-force implementation of TreeQA$^+$ is computationally expensive. TreeQA+ faces the same computational challenge as TreeQA.

- Both the number of trees and number of groupings per tree can be very large[1] in a GWA mapping.

[1]For example, the number of trees can exceed tens of thousands in a chromosome-wide association study. And there are up to 2^{2n-2} groupings that can be generated from a tree of n samples.

- Permutation tests are necessary to ensure the statistical significance of the discovered associations, which further increase the computational burden.

Since different statistical methods are used, the optimizations developed in TreeQA can not be used in TreeQA+. However, the same strategy applies, i.e., maximize the reuse of intermediate computations. A few new optimizations are developed and make TreeQA$^+$ very efficient and effective in GWA mapping, as demonstrated by extensive experiments on both simulated datasets and inbred mouse strains.

3.2 TreeQA$^+$ Method

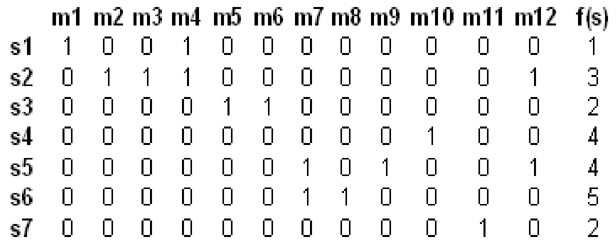
3.2.1 Preliminaries

In this chapter, we use the same set of designations and definitions (e.g., maximum compatible interval and grouping induced by a tree) as in Chapter 2. Fig. 4.2 shows an example matrix H containing 12 SNP markers, one phenotype for 7 samples and a perfect phylogenetic tree. We use this example matrix and tree as the running example in this section.

The removal of each edge partitions the tree $T_{u,v}$ into two subtrees. We use $T'_{u,v}(e_i, s_j)$ to denote the subtree rooted at node s_j (which is adjacent to edge e_i) after removing edge e_i. For example, if we remove edge e_1 in Fig. 4.2(b), we get two subtrees, $T'_{1,11}(e_1, s'_{10})$ and $T'_{1,11}(e_1, s'_8)$. And we use $T_{u,v} - T'_{u,v}(e_i, s_j)$ to represent the remaining part of the tree $T_{u,v}$ after excluding subtree $T'_{u,v}(e_i, s_j)$ and edge e_i. For example, $T'_{1,11}(e_1, s'_{10}) = T_{1,11} - T'_{1,11}(e_1, s'_8)$.

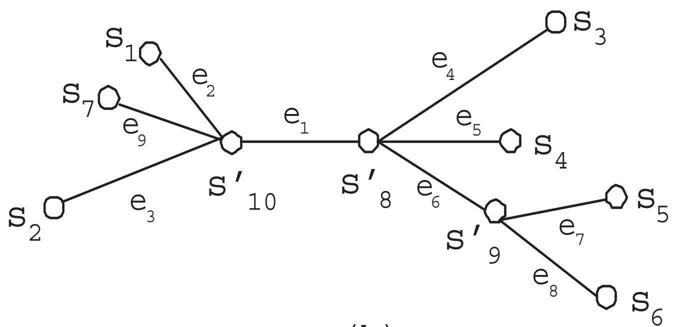
3.2.2 Association Test Incorporating Sample Correlations

A grouping (generated by removing some edges in the phylogeny tree) partitions samples into several subsets. The previous method TreeQA (Pan et al. (2009)) tests the



	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	m11	m12	f(s)
s1	1	0	0	1	0	0	0	0	0	0	0	0	1
s2	0	1	1	1	0	0	0	0	0	0	0	1	3
s3	0	0	0	0	1	1	0	0	0	0	0	0	2
s4	0	0	0	0	0	0	0	0	0	1	0	0	4
s5	0	0	0	0	0	0	1	0	1	0	0	1	4
s6	0	0	0	0	0	0	1	1	0	0	0	0	5
s7	0	0	0	0	0	0	0	0	0	0	1	0	2

(a)



(b)

Figure 3.2: A SNP dataset and a perfect phylogeny tree.

association between the partition and the phenotype values by comparing the means and variances of the partitions using a t-test or f-test. However, as discussed in Section 3.1, this approach ignores the sample correlations in each subset.

In contrast, TreeQA$^+$ incorporates sample correlations into the association test. Given a grouping, TreeQA$^+$ estimates the expected phenotype value of the root node of each subtree from the phenotype values of the leaf nodes and the topology of the subtree by utilizing Felsenstein's tree pruning algorithm (Felsenstein (1981)). Then the association between the grouping and the phenotype is measured by examining the differences between these estimated phenotype values separated by the removed edges in a Brownian motion (Nelson (1967) model).

Brownian motion (Nelson (1967)) has been used (Felsenstein (1981)) to model phenotype evolution in a population. For example, sample s_2 evolves from hypothetical sample s'_{10} in the perfect phylogeny tree in Fig. 4.2(b). If markers m_2 and m_3 (corresponding to e_3) do not contain causative mutations of the phenotype, then the difference between the phenotype values, $f(s_2)$ and $f(s'_{10})$, is normally distributed with mean 0, and variance proportional to $l(e_3)$, i.e., $(f(s_2) - f(s'_{10})) \sim N(0, k \cdot l(e_3))$, where k is the ratio between the variance and edge length. Note that the variance is an indicator of the range of estimation error.

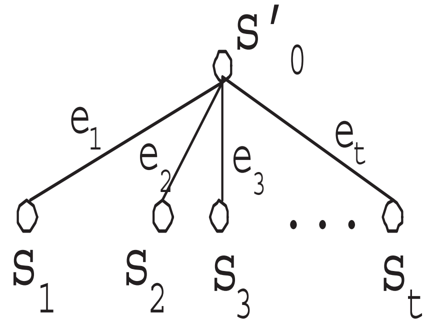


Figure 3.3: An exmaple subtree with t branches.

With this model, we can estimate the phenotype values of any hypothetical samples (intermediate nodes) using the maximum likelihood method proposed in the Felsenstein's tree pruning algorithm (Felsenstein (1981)). Given a subtree as shown in Fig. 3.3, the expected phenotype value of the root node and the estimation error (variance) are,

$$f(s'_0) = \frac{\sum_{i=1}^{t} f(s_i)/(k \cdot l(e_i))}{\sum_{i=1}^{t} 1/(k \cdot l(e_i))}, \quad v(s'_0) = \frac{1}{\sum_{i=1}^{t} 1/(k \cdot l(e_i))} \tag{3.1}$$

This estimation based on maximum likelihood was only proved for two-branch cases in (Felsenstein (1981)). However, it is easy to prove the correctness for t branches (Equation 4.5).

Given a phylogeny tree shown in Fig. 3.3, assume that we have the phenotype values of samples s_1 to s_t and the lengths of edges e_1 to e_t. We can estimate the phenotype value of s'_0 using a maximum likelihood model with Brownian motion (Nelson (1967)).

In the Brownian motion model, the phenotype difference between two nodes connected by an edge follows a normal distribution with mean 0 and variance proportional to the length of the edge. Let k be the ratio between the variance and edge length. For each sample s_i, $i = 1, 2..., t$, its phenotype value follows the following normal distribution,

$$f(s_i) \sim N(f(s'_0), k \cdot l(e_i)), \quad i = 1, 2, ..., t \tag{3.2}$$

Therefore, given the tree in Fig. 3.3, the probability that the samples have the observed phenotype values is,

$$L = \prod_{i=1}^{t} \frac{1}{\sqrt{2\pi k \cdot l(e_i)}} exp[-\frac{(f(s_i) - f(s'_0))^2}{2k \cdot l(e_i)}] \tag{3.3}$$

The log likelihood is

$$\ln L = -\sum_{i=1}^{t} (\frac{1}{2} \ln (2\pi k \cdot l(e_i)) + \frac{(f(s_i) - f(s'_0))^2}{2k \cdot l(e_i)}) \tag{3.4}$$

If we differentiate $\ln L$ with respect to $f(s'_0)$, we obtain

$$\frac{d}{d(f(s'_0))} \ln L = -\sum_{i=1}^{t} \frac{f(s'_0) - f(s_i)}{k \cdot l(e_i)} \tag{3.5}$$

In order to get the maximum likelihood estimation of $f(s'_0)$, we equate Equation 3.5 to zero and obtain,

$$f(s'_0) = \frac{\sum_{i=1}^{t} f(s_i)/(k \cdot l(e_i))}{\sum_{i=1}^{t} 1/(k \cdot l(e_i))} = \frac{\sum_{i=1}^{t} f(s_i)/l(e_i)}{\sum_{i=1}^{t} 1/l(e_i)} \tag{3.6}$$

As suggested in (Felsenstein (1981)), the estimation error (variance) of Equation 3.6 may be calculated as

$$v(s'_0) = \frac{1}{\sum_{i=1}^{t} 1/(k \cdot l(e_i))} \tag{3.7}$$

In the example in Fig. 4.2(b), we obtain a grouping consisting of two subtrees $T'_{1,11}(e_1, s'_{10})$ and $T'_{1,11}(e_1, s'_8)$ by removing edge e_1. We can assume either s'_8 evolved from s'_{10} or s'_{10} evolved from s'_8. It does not affect the following equations.

In subtree $T'_{1,11}(e_1, s'_{10})$, the expected phenotype value of node s'_{10} and the estimation error (variance) are

$$f(s'_{10}) = \frac{f(s_1)/(k_1 \cdot l(e_2)) + f(s_7)/(k_1 \cdot l(e_9)) + f(s_2)/(k_1 \cdot l(e_3))}{1/(k_1 \cdot l(e_2)) + 1/(k_1 \cdot l(e_9)) + 1/(k_1 \cdot l(e_3))} \tag{3.8}$$

$$v(s'_{10}) = \frac{1}{1/(k_1 \cdot l(e_2)) + 1/(k_1 \cdot l(e_9)) + 1/(k_1 \cdot l(e_3))} \tag{3.9}$$

In subtree $T'_{1,11}(e_1, s'_8)$, we need to apply Felsenstein's method recursively for node s'_8. We first estimate the expected phenotype value and variance of s'_9,

$$f(s'_9) = \frac{f(s_5)/(k_2 \cdot l(e_7)) + f(s_6)/(k_2 \cdot l(e_8))}{1/(k_2 \cdot l(e_7)) + 1/(k_2 \cdot l(e_8))}, \quad v(s'_9) = \frac{1}{1/(k_2 \cdot l(e_7)) + 1/(k_2 \cdot l(e_8))} \tag{3.10}$$

Then we estimate the expected phenotype value and variance of s'_8 using the estimation of $f(s'_9)$,

$$f(s'_8) = \frac{f(s_3)/(k_2 \cdot l(e_4)) + f(s_4)/(k_2 \cdot l(e_5)) + f(s'_9)/(k_2 \cdot l(e_6) + v(s'_9))}{1/(k_2 \cdot l(e_4)) + 1/(k_2 \cdot l(e_5)) + 1/(k_2 \cdot l(e_6) + v(s'_9))} \tag{3.11}$$

$$v(s'_8) = \frac{1}{1/(k_2 \cdot l(e_4)) + 1/(k_2 \cdot l(e_5)) + 1/(k_2 \cdot l(e_6) + v(s'_9))} \tag{3.12}$$

Note that because $f(s'_9)$ is an estimated value, its estimation error (variance) $v(s'_9)$ propagates to s'_8 in the estimation of $f(s'_8)$ and $v(s'_8)$. Note that in Equations 5.1 and 5.4, k_1 and k_2 are cancelled out from the numerator and denominator when estimating the phenotype values. And in Equation 5.6, k_2 is also cancelled out after we replace $v(s'_9)$ by Equation 5.4.

Under the model of Brownian motion (Nelson (1967)), the difference between the phenotype values of s'_{10} and s'_8 follows a normal distribution with mean 0 and variance proportional to the length of e_1, $(f(s'_8) - f(s'_{10})) \sim N(0, k \cdot l(e_1))$. However, since $f(s'_{10})$ and $f(s'_8)$ are estimated values with error (variance) $v(s'_{10})$ and $v(s'_8)$, we need to use the cumulative variance when we compare $f(s'_8)$ and $f(s'_{10})$. Since $f(s'_{10})$ and $f(s'_8)$ are estimated independently from two subtrees, if edge e_1 does not correspond to any causative mutation of the phenotype, we should have,

$$(f(s'_8) - f(s'_{10})) \sim N(0, k \cdot l(e_1) + v(s'_{10}) + v(s'_8)). \tag{3.13}$$

If the difference between $f(s'_8)$ and $f(s'_{10})$ deviates significantly from 0 given the variance, it is possible that some causative mutation happened on edge e_1 during the evolution which causes $|(f(s'_8) - f(s'_{10})|$ to be abnormal. Therefore, we can calculate the probability to measure the association between the grouping and phenotype, i.e., the probability that a causative mutation happened at the markers corresponding to the edge separating the groups (subtrees),

$$P(X \geq |f(s'_8) - f(s'_{10})|) = \frac{2}{\sigma\sqrt{2\pi}} \int_{-\infty}^{-|f(s'_8)-f(s'_{10})|} exp(-\frac{(u-0)^2}{2\sigma^2})du \tag{3.14}$$

where $\sigma^2 = k \cdot l(e_1) + v(s'_{10}) + v(s'_8)$ is the variance in Equation 3.13. A low probability $P(X \geq |f(s'_8) - f(s'_{10})|)$ indicates a strong association between the grouping and phenotype.

Under the model of Brownain motion, the variances are proportional to the edge lengths. However, the ratios between the variances and edge lengths may be different in different subtrees. In Equations 5.2, 3.12, and 3.13, we assume that the ratio is a constant within each subtree if there is no causative mutation. We use the average ratio in each subtree as its estimated ratio. Let $D(s_i, s_j)$ denote the sum of edge lengths from node s_i to s_j in the tree. Then the estimated values of k_1 and k_2 are,

$$k_1 = \frac{(f(s_1) - f(s'_{10}))^2 + (f(s_7) - f(s'_{10}))^2 + (f(s_2) - f(s'_{10}))^2}{D(s_1, s'_{10}) + D(s_7, s'_{10}) + D(s_2, s'_{10})} \tag{3.15}$$

$$k_2 = \frac{(f(s_3) - f(s'_8))^2 + (f(s_4) - f(s'_8))^2 + (f(s_5) - f(s'_8))^2 + (f(s_6) - f(s'_8))^2}{D(s_3, s'_8) + D(s_4, s'_8) + D(s_5, s'_8) + D(s_6, s'_8)} \tag{3.16}$$

And the ratio on edge e_1 that connects the two subtrees is estimated by the average ratios of the two subtrees, i.e., $k = (k_1 + k_2)/2$. Note that the expected values of $f(s'_{10})$ and $f(s'_8)$ do not depend on k_1 and k_2 in Equations 5.1 and 5.6. Only the variances are influenced by the values of k_1 and k_2.

The sample correlation implied by the tree topology is incorporated into the calculation of the probability in Equation 3.14. Therefore, TreeQA$^+$ can avoid the bias caused by sample correlations in association analysis. For example, given the three trees with branch lengths labelled in Fig. 4.1, let's revisit the grouping formed by removing the middle edge. The groupings have probability scores $P = 0.841$, $P = 0.829$, and $P = 0.479$ in trees (a), (b), and (c) respectively. The high probability scores for trees (a) and (b) suggest that the associations are weak in both trees. The lower probability score for tree (c) indicates a stronger association between the grouping in tree (c) and the phenotype. Note that, by taking into account the sample correlations, the grouping in tree (b) no longer shows strong association with the phenotype.

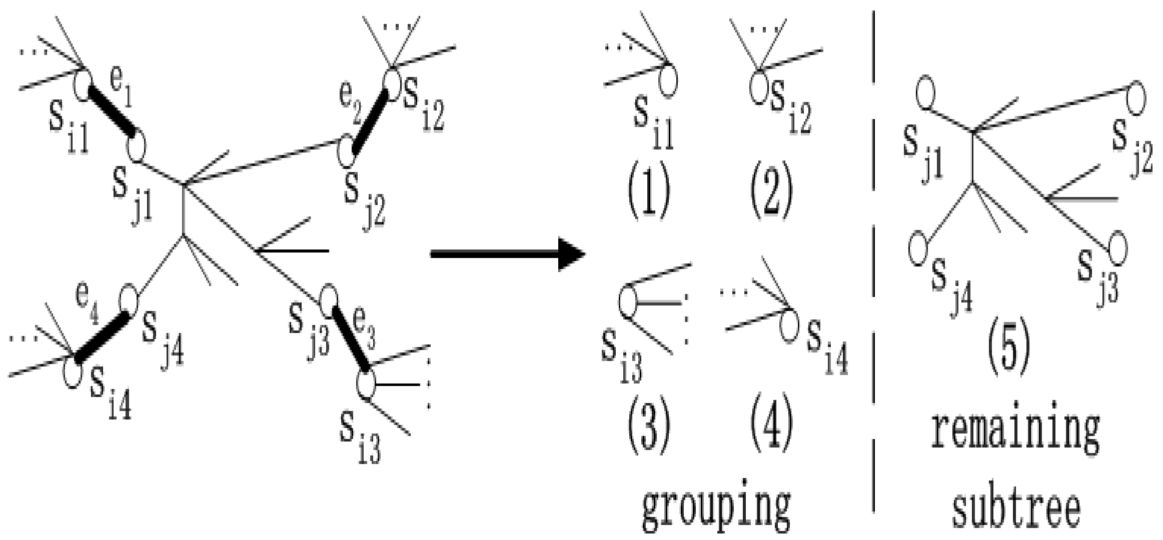


Figure 3.4: General case groupings.

General Case.

The above procedure can also be applied to the general case where we have more than two subtrees in a grouping. In Fig. 3.4, we have a grouping formed by four subtrees, $T'(e_1, s_{i_1})$, $T'(e_2, s_{i_2})$, $T'(e_3, s_{i_3})$ and $T'(e_4, s_{i_4})$. The procedure is:**(a)** We estimate the phenotype values, $f(s_{i_1})$,$f(s_{i_2})$, $f(s_{i_3})$ and $f(s_{i_4})$, using subtrees (1) to (4) in Fig. 3.4 respectively by Equation 4.5 (applying recursively if necessary); **(b)** We use the remaining subtree (subtree (5) in Fig. 3.4) to estimate $f(s_{j_1})$,$f(s_{j_2})$, $f(s_{j_3})$ and $f(s_{j_4})$ in the same way by treating each of s_{j_1}, s_{j_2}, s_{j_3}, and s_{j_4} as the root respectively; **(c)** The ratios, k_i, of the subtrees are estimated using Equations 3.15 and 3.16; **(d)** Then we calculate probability,

$$P = \prod_{q=1}^{4} P(X \geq |f(s_{i_q}) - f(s'_{j_q})|) \tag{3.17}$$

A low probability indicates a strong association between the grouping and phenotype.

Since we may have a large number of trees and a large number of groupings induced by each tree in a GWA mapping, we apply a **permutation test** on each grouping to determine whether the association between the grouping and phenotype is indeed significant statistically.

1. We calculate a base P_0 probability on the given grouping by Equation 5.3.

2. We permute the phenotype values of the samples in all subtrees in the grouping (e.g. subtrees (1) to (4) in Fig. 3.4) $\boldsymbol{N}$ times. We calculate a probability P_i for each permutation by Equation 5.3.

3. Let n be the number of times when P_i is less than P_0. The finial **significance score** of the grouping is,

$$S = \frac{n}{N} \tag{3.18}$$

3.2.3 Algorithm Implementation and Optimizations

Same as TreeQA, TreeQA$^+$ takes two major steps:

1. We identify maximal compatible regions in the genome and construct the perfect phylogenies of the regions.

2. For each phylogeny tree, we enumerate all induced groupings and compute the significance score of the association between each grouping and the phenotype using Equations 5.3 and 3.18. The most significant score for each tree is reported.

TreeQA+ uses the same procedure to identify maximal compatible regions as described in Chapter 2. However, TreeQA+ enumerates groupings and examines association in a different way.

As we discussed in Section 2.3, a grouping is formed by a set of non-overlapping subtrees, $T'_{u,v}(e_i, s_j)$. Given a local perfect phylogeny, TreeQA$^+$ enumerates all implied groupings by enumerating all combinations of non-overlapping subtrees via a recursive procedure. In order to explore the search space systematically, we first order all subtrees $T'_{u,v}(e_i, s_j)$ of $T_{u,v}$. Each subtree $T'_{u,v}(e_i, s_j)$ can be uniquely identified by the removed edge e_i and the root node s_j. If we define a total order of the edges and nodes, we can generate a total order of the subtrees, i.e., $T'_{u,v}(e_{i_1}, s_{j_1}) < T'_{u,v}(e_{i_2}, s_{j_2})$ if $e_{i_1} < e_{i_2}$ and $s_{j_1} < s_{j_2}$. For example, given the tree in Fig. 4.2(b), we define the order of edges and nodes as $e_1 < e_2 < ... < e_9$ and $s_1 < s_2 < ... < s'_9 < s'_{10}$. Then the subtrees have the following order, $T'_{1,11}(e_1, s'_8), T'_{1,11}(e_1, s'_{10}), T'_{1,11}(e_2, s_1), T'_{1,11}(e_2, s'_{10})$,$T'_{1,11}(e_3, s_2), T'_{1,11}(e_3, s'_{10})$, $T'_{1,11}(e_4, s_3), T'_{1,11}(e_4, s'_8)$...

With an ordered list of subtrees, we can use any standard subset enumeration method (Loughry et al. (2002)) to enumerate all combinations of subtrees. Combinations containing overlapping subtrees are excluded during the enumeration to enforce the non-overlapping constraint. For example, with the above list of subtrees, if we enumerate the subtrees in lexicographical order, TreeQA$^+$ examines the combinations of subtrees in the following order: $\{T'_{1,11}(e_1, s'_8), T'_{1,11}(e_1, s'_{10})\}$, $\{T'_{1,11}(e_1, s'_8), T'_{1,11}(e_2, s_1)\}$, $\{T'_{1,11}(e_1, s'_8), T'_{1,11}(e_2, s_1), T'_{1,11}(e_3, s_2)\}$.... Combination $\{T'_{1,11}(e_1, s'_8), T'_{1,11}(e_1, s'_{10}), T'_{1,11}(e_2, s_1)\}$ would have been the second one in the lexicographical order but is excluded because $T'_{1,11}(e_1, s'_{10})$ and $T'_{1,11}(e_2, s_1)$ overlap. This procedure continues until all combinations of non-overlapping subtrees are explored. In practice, subtrees containing too few samples may be excluded from the list of subtrees because of their lack of statistical significance.

For each grouping, TreeQA$^+$ computes the significance score of its association with the phenotype via a permutation test and using Equations 3.14, 5.3 and 3.18. In the end, TreeQA$^+$ reports the most significant score of the tree.

In GWA mapping, we can have a large number of trees and groupings. We maximize the reuse of intermediate computations to make TreeQA$^+$ efficient in GWA mapping.

1. The Felsenstein's tree pruning method (Felsenstein (1981)) is used to estimate the phenotype values at the root node of every subtree. Instead of invoking this method for each subtree separately, TreeQA$^+$ re-designs the method implementation such that the estimated phenotype values of the root nodes of all subtrees are calculated through two scans of the tree only.

2. In TreeQA$^+$, we apply a permutation test on each induced grouping. We randomly permute the phenotype values of leaf nodes in the subtrees in the grouping. However, the phenotype values in the remaining part of the tree retain the same. Therefore, any computation on the remaining part of the tree can be reused in the permutation test of the grouping. For example, given the grouping in Fig. 3.4, we only permute the phenotype values in subtrees, $T(e_1, s_{i_1})$, $T(e_2, s_{i_2})$, $T(e_3, s_{i_3})$ and $T(e_4, s_{i_4})$. The phenotype values in the remaining part of the tree (subtree (5) in Fig. 3.4) retain the same. Thus, the estimated phenotype values of the nodes s_{j_1}, s_{j_2}, s_{j_3} and s_{j_4} do not change and can be reused in the entire permutation test of the grouping.

3.3 Experimental Results

We compare TreeQA$^+$ with other algorithms representing single-marker, haplotype, and local phylogeny-based association mapping methods on both simulated data and inbred mouse strains:

1. **SMA**, our implementation of the Single Marker Association algorithm (Pe'er et al. (2006); Akey et al. (2001)).
2. **HAM**, our implementation of the Haplotype Association Mapping algorithm (McClurg et al. (2006)) that slides a 3-SNP window through the genome.
3. **HapMiner** (Wang and Paigen (2005)).
4. **TreeLD** (Zollner and Pritchard (2005)).
5. our previous method **TreeQA** Pan et al. (2009).

SMA and HAM use one-way ANOVA test with permutation test. We do not compare with QHPM (Onkamo et al. (2002)), Blossoc (Mailund et al. (2006)) and TreeDT (Sevon et al. (2006)) because QHPM is not scalable to large data, while Blossoc and TreeDT only handle categorical phenotypes.

3.3.1 Experiments on Simulated Data

We use Coasim (Mailund et al. (2005)) to simulate 1000 sequences and 10,000 SNP markers with scaled recombination rate $\rho = 400$ that corresponds to 10 cM roughly. The SNP markers are placed uniformly at random over the sequences.

We randomly select SNP markers to be the causative loci with one, two and three causative mutations. Causative SNPs are added in the same way as described in Chapter 2.

SMA, HAM and HapMiner output the top one scoring locus as a point estimation of the causative locus, while TreeQA and TreeQA$^+$ output the top one compatible region. We measure the distance (in cM) from the top one scoring locus or the center of the top one region to the causative SNP (or the average distance to every causative SNP) to compare the effectiveness of the algorithms. We call this distance the **Prediction error**. HapMiner is unable to process 10,000 markers in reasonable time. Thus, we only use the first 1,000 markers of each sequence in the experiments of HapMiner.

The comparison of SMA, HAM, HapMiner, TreeQA and TreeQA$^+$ is shown in Fig. 3.5 (a). The x-axis represents the prediction error to the causative locus and the y-axis represents the percentage of causative loci which are found with prediction error less than x. In all three cases, TreeQA$^+$ outperforms TreeQA and all other algorithms. TreeQA$^+$ has smaller prediction error than TreeQA because sample correlations are incorporated into TreeQA$^+$.

TreeLD is a phylogeny-based method which is very time-consuming. We build a smaller dataset using 36 samples and 20 SNPs from the simulated data for performance comparison. A one-mutation causative locus is selected. Because of the small number of SNPs in the data, TreeQA and TreeQA$^+$ generate a compatible region around each SNP which contains up to five SNPs. TreeLD takes two hours to analyze this small dataset while both TreeQA and TreeQA$^+$ finish in seconds. In Fig. 2.5 (b), the x-axis represents the SNP positions in the genome and the y-axis represents the significance scores. The vertical line indicates the causative locus. The peak detected by TreeQA$^+$ is closer to the causative locus than those of TreeQA and TreeLD. TreeLD also identifies another spurious peak.

3.3.2 Experiments on Mouse Genotype Data

We used a set of mouse genotypes which combines real and imputed data[2] Szatkiewicz et al. (2008) (NCBI Build 36) from the Jackson Laboratory. It consists of 55 samples that have common ancestral relationships Yang et al. (2007) and over 7 million SNP markers distributed over 20 chromosomes.

The high density lipoprotein cholesterol (HDL-C) levels in blood are used as the test phenotypes, downloaded from the Mouse Phenome Database[3]. Several HDL-C datasets are available, each of which was collected under different conditions, and is thus treated as a separate phenotype. Some candidate genes that may play a role in regulating HDL-C levels are reported in (Wang and Paigen (2002)).

We apply SMA, HAM, TreeQA and TreeQA$^+$ on the data and examine the distance of the top peak reported by each algorithm to the loci of those candidate genes. Both TreeQA and TreeQA$^+$ detect top peaks near the loci for over 10 of the candidate genes (Wang and Paigen (2002)) while the top peaks reported by SMA and HAM are often far from the loci of any genes. Moreover, some candidates genes are only detected by TreeQA$^+$, such as *abcb11* (in Fig. 3.6(a)) and *lipg*. And for some candidate genes such as *apoc2* and *apoe* (in Fig. 3.6(b)), the peaks detected by TreeQA$^+$ are more significant and closer to the loci than the peaks of TreeQA. Due to space limitation, the results of SMA are omitted in Fig. 3.6 because they are always worse than or similar to the results of HAM.

Fig. 3.7 plots the perfect phylogenies at the two peak points (compatible regions from 18875865 to 18880075 and from 16194875 to 16195002 (base)) found by TreeQA$^+$ and TreeQA in Fig. 3.6(b). Branch lengthes of the two trees are in the same scale. The phenotype values of the samples are in parentheses. Samples with unknown phenotype values are omitted from the tree. The peak detected by TreeQA deviates away from

[2]http://cgd.jax.org/ImputedSNPData/imputedSNPs.htm

[3]http://phenome.jax.org/pub-cgi/phenome/mpdcgi?rtn=meas/catlister/req=Cblood+lipids

the *apoe/apoc2* loci because the sample correlations represented in the phylogeny at the peak location are very high. Ignoring sample correlations misleads TreeQA to report a spurious association. HAM (and also SMA) fail to identify the locus because they only examine sample groupings generated from single SNPs or 3-SNP windows, which do not include the grouping created by removing two edges in Fig. 3.7(a).

TreeQA$^+$ takes about 10 minutes to analyze each chromosome (containing around 40000 SNPs on average) which is similar to the runtime of TreeQA. SMA and HAM take slightly less time. All these methods use 100000 permutations in the permutation tests.

3.4 Discussion

In this chapter, we present the TreeQA$^+$ method, a local phylogeny-based GWA mapping method which incorporates sample correlations. The model of Brownian motion Nelson (1967) and the Felsenstein's tree pruning method Felsenstein (1981) are utilized in TreeQA$^+$ to incorporate sample correlations. By careful algorithm design and implementation, we reduce the high computational cost of TreeQA$^+$ and make it efficient for genom-wide analysis. We demonstrate that:

1. TreeQA$^+$ outperforms single-maker and haplotype-based methods because it examines all groupings induced by the phylogeny trees instead of only groupings generated from single SNPs or 3-SNP windows.

2. TreeQA$^+$ avoids the detection of spurious peaks in other phylogeny-based methods such as TreeQA by taking into account sample correlations.

3. TreeQA$^+$ has comparable runtime performance to other efficient GWA mapping methods.

Therefore, TreeQA$^+$ is supreme to all previous methods in GWA mapping, in terms of accuracy, efficiency and robustness.

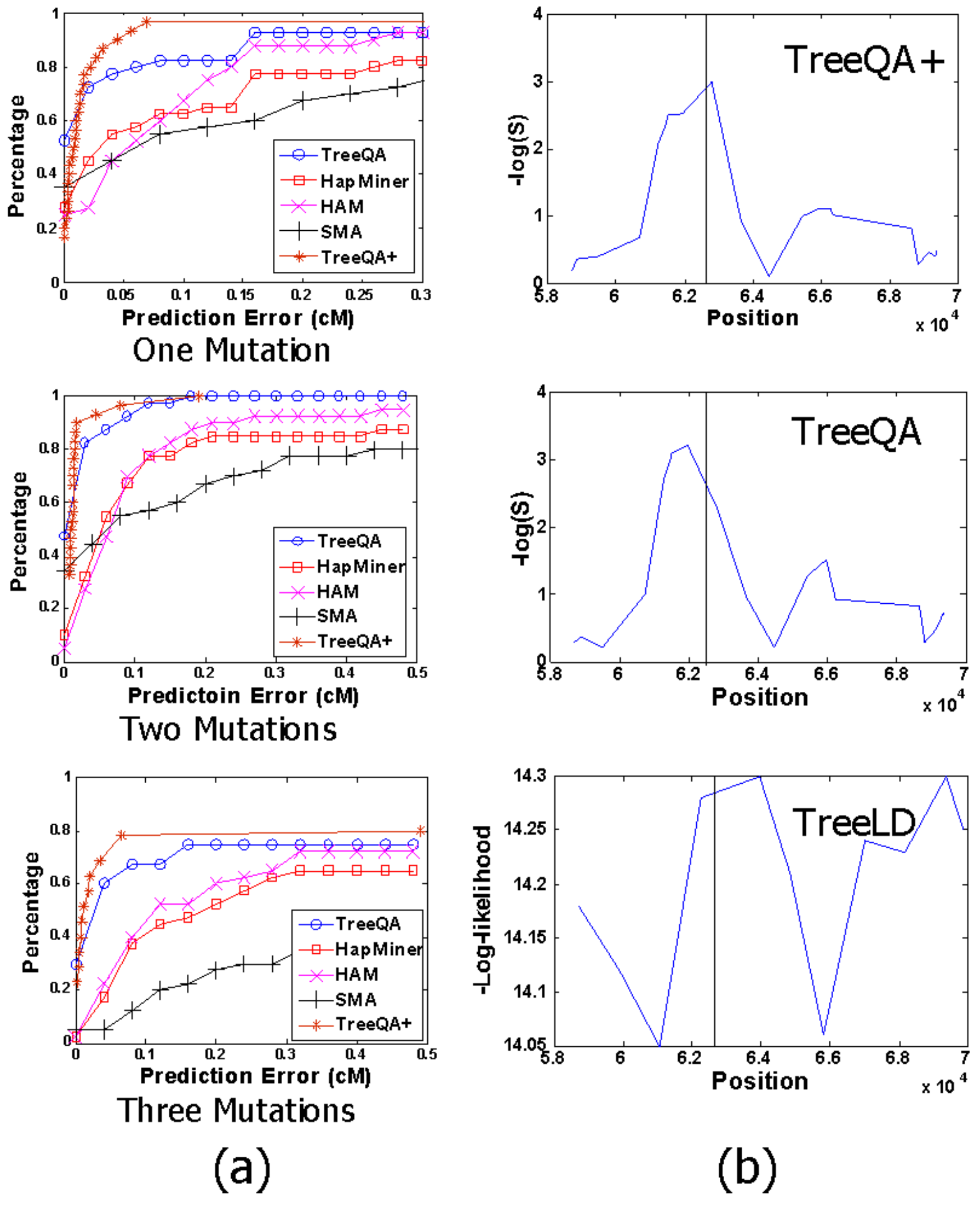
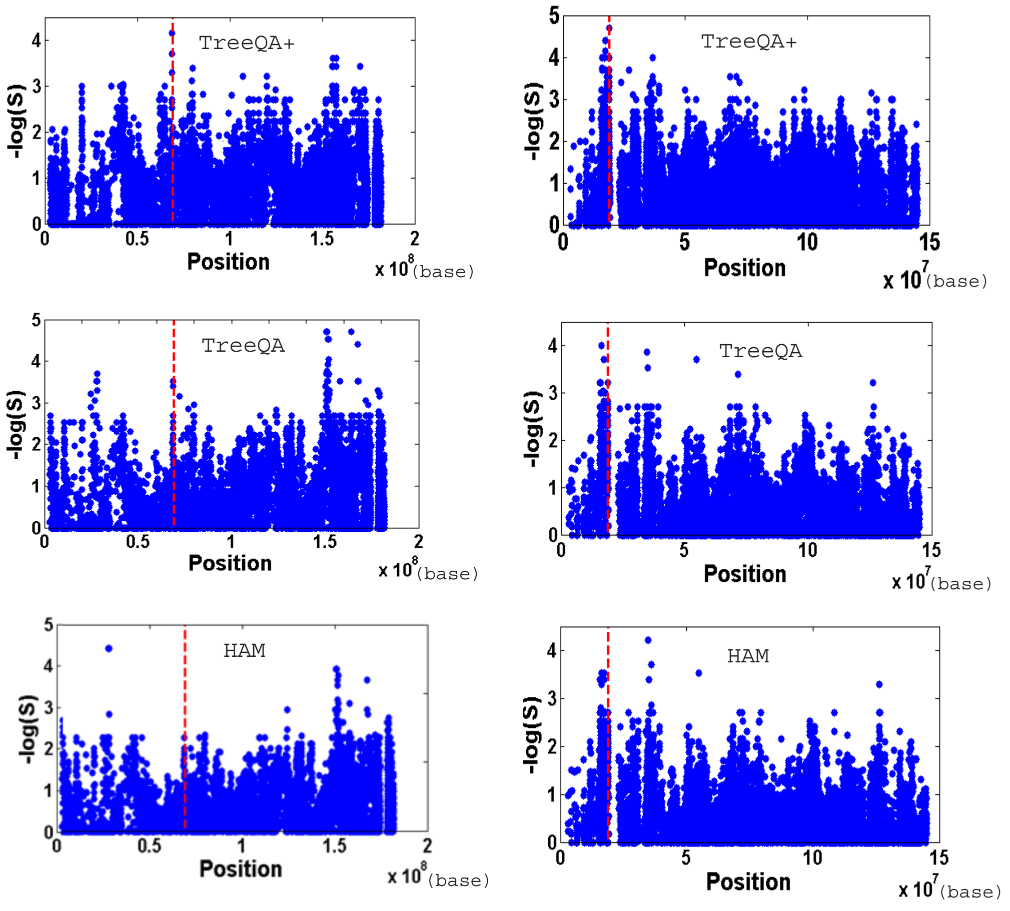


Figure 3.5: Comparison of SMA, HAM, HapMiner, TreeLD, TreeQA and TreeQA$^+$ on the simulated data.



(a) Gene: *Abcb11* on chromosome 2, 69039121–69143453 (base), NCBI Build 36
Phenotype: Paigen 2, baseline of HDL cholesterol (female)

(b) Gene: *Apoc2/Apoe* on chromosome 7, 18830106–18836463(base) and 18854795–18857574(base), NCBI Build 36.
Phenotype: Paigen 2, HDL cholesterol level after 17 wks (male)

Figure 3.6: Compare HAM, TreeQA and TreeQA$^+$ on the mouse genotype data

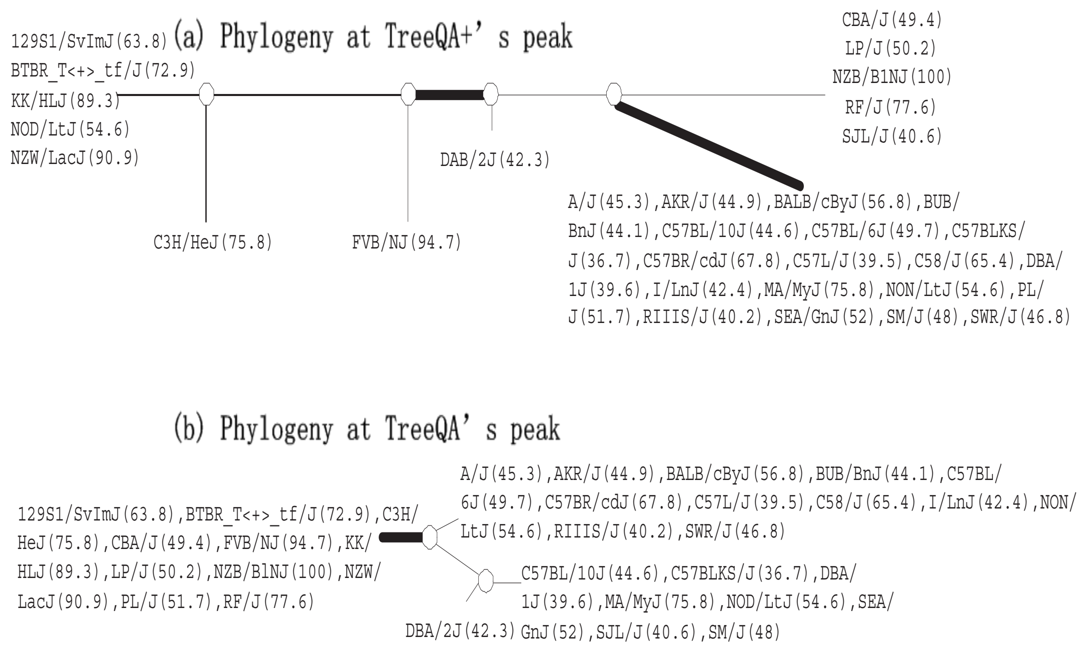

(a) Phylogeny at TreeQA+' s peak
129S1/SvImJ(63.8)
BTBR_T<+>_tf/J(72.9)
KK/HLJ(89.3)
NOD/LtJ(54.6)
NZW/LacJ(90.9)
C3H/HeJ(75.8)
FVB/NJ(94.7)
DAB/2J(42.3)
CBA/J(49.4)
LP/J(50.2)
NZB/B1NJ(100)
RF/J(77.6)
SJL/J(40.6)
A/J(45.3),AKR/J(44.9),BALB/cByJ(56.8),BUB/BnJ(44.1),C57BL/10J(44.6),C57BL/6J(49.7),C57BLKS/J(36.7),C57BR/cdJ(67.8),C57L/J(39.5),C58/J(65.4),DBA/1J(39.6),I/LnJ(42.4),MA/MyJ(75.8),NON/LtJ(54.6),PL/J(51.7),RIIIS/J(40.2),SEA/GnJ(52),SM/J(48),SWR/J(46.8)
(b) Phylogeny at TreeQA' s peak
129S1/SvImJ(63.8),BTBR_T<+>_tf/J(72.9),C3H/HeJ(75.8),CBA/J(49.4),FVB/NJ(94.7),KK/HLJ(89.3),LP/J(50.2),NZB/BlNJ(100),NZW/LacJ(90.9),PL/J(51.7),RF/J(77.6)
A/J(45.3),AKR/J(44.9),BALB/cByJ(56.8),BUB/BnJ(44.1),C57BL/6J(49.7),C57BR/cdJ(67.8),C57L/J(39.5),C58/J(65.4),I/LnJ(42.4),NON/LtJ(54.6),RIIIS/J(40.2),SWR/J(46.8)
C57BL/10J(44.6),C57BLKS/J(36.7),DBA/1J(39.6),MA/MyJ(75.8),NOD/LtJ(54.6),SEA/GnJ(52),SJL/J(40.6),SM/J(48)
DBA/2J(42.3)

Figure 3.7: The two phylogenies at the peaks found by TreeQA and TreeQA$^+$ respectively in Figure 3.6(b)

Chapter 4

TreeNL: Expand TreeQA to Association/Correlation Analysis in Data Mining

4.1 Introduction

In Chapters 2 and 3, I discussed the phylogeny-based genome-wide association mapping methods, TreeQA and TreeQA+. The idea of TreeQA can be expanded and applied to correlation clustering in high dimensional data.

In recent years, high dimensional data arise frequently in various applications such as text mining, business data analyzing and bio-informatics. High dimensionality poses challenges to classical data analyzing algorithms such as clustering. Instead of forming clusters in the full feature space, the correlation among a set of data objects may only be identified in a feature subspace. Moreover, objects in different clusters can be correlated in different feature subspaces. Therefore, the problem of detecting correlation clusters in the subspaces of high dimensional data has gained increasing interests.

Several algorithms (Böhm et al. (2004); Aggarwal and Yu (2000); Achtert et al. (2006); Zhang et al. (2008)) have been proposed to detect clusters of data objects which are linearly correlated in feature subspace. ORCLUS (Aggarwal and Yu (2000)) and 4C

(Aggarwal and Yu (2000)) both use the concept of micro-clusters to detect correlated feature subspaces. They first partition objects into micro-clusters using k-means based or density-based clustering approach. Then micro-clusters which have similar orientations are merged to form larger clusters. The CARE (Zhang et al. (2008)) algorithm uses a different strategy. It explicitly explores the feature subspaces, and finds the set of objects that are linearly correlated in each feature subset.

As discussed in (Tung et al. (2005)), features may exhibit nonlinear correlations in real data such as physical data and gene expression data. Therefore, CURLER (Tung et al. (2005)) was proposed to identify nonlinear correlation clusters. CURLER utilizes an EM (expectation maximization) based fuzzy clustering algorithm to form micro-clusters of objects. It then adopts an interactive top-down approach to find nonlinear correlation clusters.

Except for CARE, the algorithms discussed above (ORCLUS, 4C, CURLER) all generate a strict partition of the data objects, that is, objects are not allowed to be shared by multiple correlation clusters. However, cases have been observed in real data where data objects could behave differently in multiple feature subspaces. For example, in CARE (Zhang et al. (2008)), subsets of biological samples were found to have different linear correlations in multiple gene subsets.

An example data is presented in Figure 4.1. The data matrix contains 300 objects and 7 features. There are three correlation clusters in the data, clusters 1, 2 and 3. The objects in cluster 1 and 2 have linear correlations in feature subsets $\{1, 2, 3\}$ and $\{5, 6, 7\}$ respectively. And the objects in cluster 3 have a non-linear correlation in feature subset $\{3, 4, 5\}$. There is no intersection of objects between clusters 1 and 3. But cluster 2 shares some common objects with each of them.

Algorithms like ORCLUS (Aggarwal and Yu (2000)) and 4C (Böhm et al. (2004)) are able to find cluster 1 in Figure 4.1. CURLER (Tung et al. (2005)) is able to find both clusters 1 and 3. However, all of them will miss cluster 2 because of the overlap-

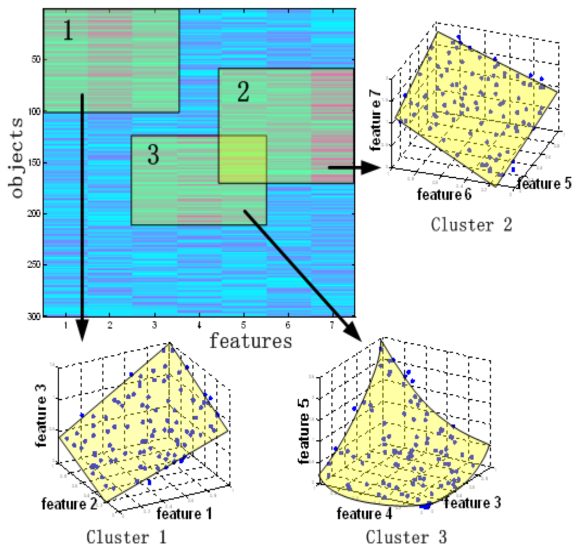


Figure 4.1: Example Data

ping of data objects between these clusters. While CARE (Zhang et al. (2008)) allows arbitrary overlapping between clusters, it is not able to find cluster 3 which is a nonlinear correlation cluster. Furthermore, CARE is not designed for detecting correlation clusters containing a small number of objects. CARE misses cluster 2 according to our experiments in Section 6.5.

In this chapter, we address these problems and develop a nonlinear correlation clustering algorithm which expands the TreeQA algorithm. The algorithm allows arbitrary overlapping between clusters. A brute-force method is to explicitly enumerate all combinations of feature subsets and data object subsets and check for possible correlation in each combination. However, this method poses two computational challenges.

1. Enumerating all possible subsets of data objects is computationally infeasible. Datasets generated in user recommendation systems, web log and other applications can easily contain thousands or millions of data objects.

2. Checking the correlation in each combination is also expensive. Even though we can utilize methods like micro-clusters and PCA (principle component analysis)

(Böhm et al. (2004); Tung et al. (2005)), the procedure will still be very time consuming considering the total number of combinations.

In this chapter, we introduce a nonlinear correlation clustering algorithm, TreeNL, to meet these challenges. TreeNL enumerates feature subspaces. By doing so, clusters of data objects which have linear/nonlinear correlations in each feature subset can be detected independently. Therefore, data objects are allowed to participate multiple correlation clusters. TreeNL utilizes tree hierarchies of data objects and BIC (Bayesian information criterion) to detect correlation in each feature subspace.

1. For each feature subset, TreeNL organizes the data objects into a tree hierarchy according to their similarities in the feature subspace. Each node in the tree represents a subset of data objects. The tree hierarchy provides both effectiveness and efficiency in correlation detection: 1) The tree hierarchy captures the distribution of the objects in the feature subspace and hence is able to reveal non-linear correlations in the feature subspace; 2) The tree hierarchy provides a multi-level resolution to examine the data objects. Instead of enumerating all possible subsets of data objects, TreeNL enumerates all possible groupings of objects implied by the tree hierarchy. By controlling the minimum size of the clusters, TreeNL ensures the statistical significance of the detected correlation clusters and is able to explore the object subspaces efficiently.

2. Instead of using PCA and expensive matrix operations, TreeNL adopts BIC (Bayesian information criterion) to measure correlation between features. The fast calculation of BIC enables TreeNL to handle large datasets efficiently.

We conducted extensive experiments on both real and synthetic datasets. The results demonstrate that TreeNL can accurately and efficiently find nonlinear correlation clusters with arbitrary overlapping.

4.2 Related Work

The problem of finding correlation clusters has been extensively studied.

ORCLUS (Aggarwal and Yu (2000)) is a partition-based linear correlation algorithm. It adopts a similar procedure as the traditional k-means clustering algorithm. Instead of calculating the Euclidian distance from each object to its cluster centroid in the full feature space, ORCLUS measures the distance in the feature subspace corresponding to each cluster. By doing so, ORCLUS can find linear correlation clusters in arbitrarily oriented feature subspaces.

4C (Böhm et al. (2004)) is a combination of density-based clustering algorithm (DBSCAN) and principle component analysis (PCA). It adopts a novel correlation similarity measurement which considers the orientation of the feature subspace. Linear correlation clusters are formed by first identifying core data objects in dense areas. Then the clusters are expended via the reachability between the data objects.

CARE (Zhang et al. (2008)) detects correlation clusters by explicitly exploring feature subspaces. For each combination of features, it adopts PCA to identify the weak eigenvectors of the feature subspace. The existence of these weak eigenvectors indicates the correlation between the features and also provides quantitative information on the linear dependencies between the correlated features (Achtert et al. (2006)). For each feature subset, CARE uses heuristics to remove a few data objects to refine the weak eigenvectors instead of explicitly exploring all the object subsets. Thus, in order to identify correlated feature subsets, CARE requires that a large portion of data objects support the linear dependencies. Consequently, CARE is not suitable for detecting minor correlation clusters in the dataset.

CURLER (Tung et al. (2005)) aims at finding nonlinear correlation clusters. It first utilizes an EM-based fuzzy clustering algorithm to partition the data objects into tiny clusters, i.e., micro-clusters. Then CURLER merges these micro-clusters according to the co-sharing level and forms nonlinear correlation clusters. The EM-based fuzzy clustering algorithm enables CURLER to assign each data object to multiple micro-clusters with different probabilities. However, after the micro-clusters are merged into larger clusters, each data object is assigned to a unique cluster according to the object's highest membership probability.

Except for CARE, the algorithms reviewed above (ORCLUS, 4C and CURLER) do not allow objects be shared by multiple correlation clusters, and thus produce a strict partition of the data objects.

Algorithms (Aggarwal et al. (1999); Agrawal et al. (1998); Hinneburg and Keim (1999); Liu and Wang (2003); Procopiuc et al. (2002)) were proposed to find clusters in axis-parallel feature subspaces. These algorithms are not able to find local, arbitrarily oriented correlation clusters.

4.3 Preliminaries

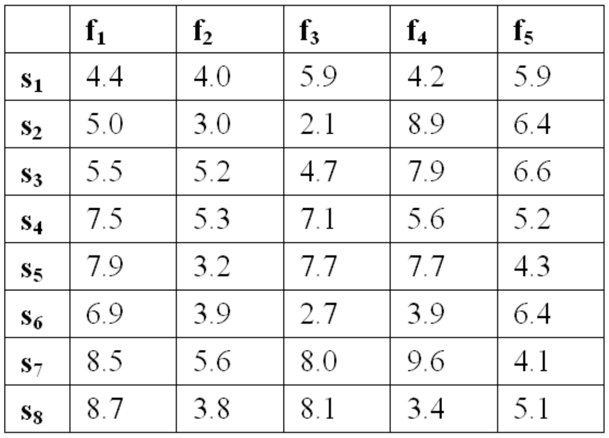
In this section, we introduce our notations (Table 4.1).

Let $H = S \times F$ be a data matrix consisting of n data objects in m-dimension. S represents the data object set (samples), $S = \{s_1, s_2, ..., s_n\}$. F represents the feature set, $F = \{f_1, f_2, ..., f_m\}$[1]. Given a data object s_i, $f_j(s_i)$ represents its value on feature f_j. An example data matrix of 8 objects and 5 features is shown in Figure 4.2.

4.3.1 Tree Hierarchy

TreeNL utilizes tree hierarchies to organize all data objects in feature subspaces. Given a feature subset F', $F' \subseteq F$, we denote its corresponding tree as $T^{F'}$. TreeNL uses a threshold, $\mathbf{g_{min}}$, to control the minimum number of objects at each node of the tree.

[1]In this chapter, H is representing any general datasets, not only binary SNP data. Therefore, we use F to represent the feature set. It is not as Chapter 2 that we use M to represent the marker set



	f_1	f_2	f_3	f_4	f_5
s_1	4.4	4.0	5.9	4.2	5.9
s_2	5.0	3.0	2.1	8.9	6.4
s_3	5.5	5.2	4.7	7.9	6.6
s_4	7.5	5.3	7.1	5.6	5.2
s_5	7.9	3.2	7.7	7.7	4.3
s_6	6.9	3.9	2.7	3.9	6.4
s_7	8.5	5.6	8.0	9.6	4.1
s_8	8.7	3.8	8.1	3.4	5.1

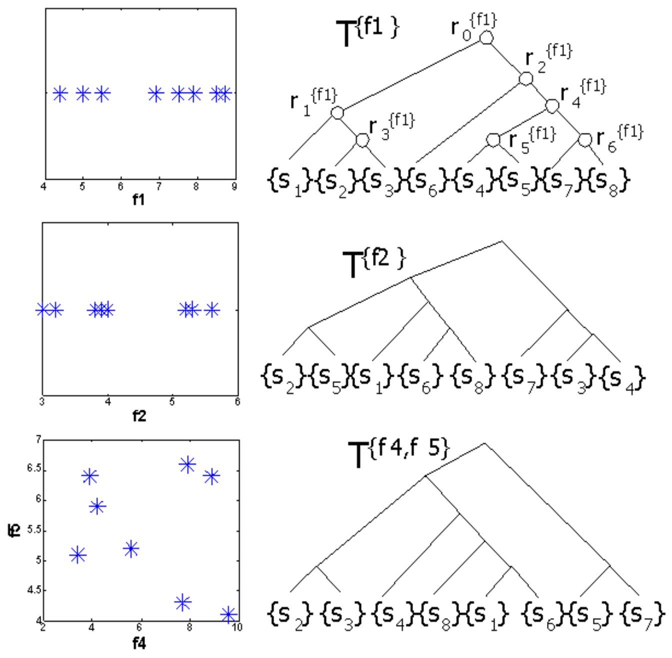
Figure 4.2: An example 8×5 data matrix

Definition 4.3.1. Tree Hierarchy: Given a feature subset F' and g_{min}, the corresponding tree $T^{F'}$[2] consists of a set of tree nodes $R^{F'} = \{r_0^{F'}, r_1^{F'}, ..., r_w^{F'}\}$. $r_0^{F'}$ is the root node of tree $T^{F'}$. Each node $r_i^{F'}$ represents a data object subset, $\mathsf{S}(r_i^{F'})$.

- $\forall \mathsf{S}(r_i^{F'})$, $|\mathsf{S}(r_i^{F'})| \geq g_{min}$.
- Given any two leaf nodes $r_i^{F'}$ and $r_j^{F'}$, their object subsets are disjoint, $\mathsf{S}(r_i^{F'}) \cap \mathsf{S}(r_j^{F'}) = \emptyset$.
- Given an intermediate node $r_i^{F'}$, $\mathsf{S}(r_i^{F'}) = \bigcup_j \mathsf{S}(r_j^{F'})$, where $\{r_j^{F'}\}$ are the leaf nodes in the subtree of $r_i^{F'}$.
- $\mathsf{S}(r_0^{F'}) = S$.

In general, different trees may be constructed for a given feature subset. In this chapter, given F', we use $T^{F'}$ to represent the tree hierarchy constructed by TreeNL via hierarchical clustering. Given the example data in Figure 4.2, three trees, $T^{\{f_1\}}$, $T^{\{f_2\}}$ and $T^{\{f_4,f_5\}}$ constructed by TreeNL with $g_{min} = 1$ are plotted in Figure 4.3.

[2]In Chapter 2, trees are represented by $T_{u,v}$ because they are inferred from compatible regions $C_{u,v}$. In this chapter, trees are constructed on feature subsets F', thus, we use $T^{F'}$ to represent the corresponding tree



T{f1}
r0{f1}
r1{f1}
r2{f1}
r3{f1}
r4{f1}
r5{f1}
r6{f1}
{s1}{s2}{s3}{s6}{s4}{s5}{s7}{s8}
f1
T{f2}
{s2}{s5}{s1}{s6}{s8}{s7}{s3}{s4}
f2
T{f4,f5}
{s2}{s3}{s4}{s8}{s1}{s6}{s5}{s7}
f4
f5

Figure 4.3: Trees $T^{\{f_1\}}$, $T^{\{f_2\}}$ and $T^{\{f_4,f_5\}}$ of the data matrix in Figure 4.2, with $g_{min} = 1$

In Figure 4.3, we only give the object subsets represented by the leaf nodes. Since $g_{min} = 1$, there is only one object in each subset. The intermediate node $r_4^{\{f_1\}}$ in tree $T^{\{f_1\}}$ represents object subset $\{s_4, s_5, s_7, s_8\}$.

We usually set g_{min} to be much larger than 1. Therefore, TreeNL adopts a top-down hierarchical clustering approach to generate tree $T^{F'}$ for a given feature subset F' instead of using a bottom-up approach. Tree $T^{F'}$ organizes data objects according to their similarities in feature subspace F'. We may use Euclidian distance or any other similarity/distance functions. Data objects which are close to each other in feature subspace F' are grouped into the same subtree of $T^{F'}$. Details of the tree construction algorithm are discussed in Section 4.5.

4.3.2 Groupings of Objects Indicated by a Tree

TreeNL partitions (clusters) data objects into subsets implied by the tree hierarchy. In Chapter 2, we defined the groupings indicated by a tree. In this section, we define this concept more formally. Given a tree $T^{F'}$, a grouping indicated by $T^{F'}$ consists of a set of *disjoint* data object subsets which are represented by tree nodes of $T^{F'}$.

Definition 4.3.2. Grouping of Objects Indicated by a Tree Hierarchy: Let $G_i^{F'}$ denote a grouping of objects indicated by tree $T^{F'}$, $F' \subseteq F$. $G_i^{F'} = \{ \mathsf{S}(r_j^{F'}), j = \{1, ..., l\} | \forall u, v \in \{1, ..., l\}, \mathsf{S}(r_u^{F'}) \cap \mathsf{S}(r_v^{F'}) = \emptyset\}$.

In the rest of this paper, we use term 'grouping' to denote 'grouping of data objects' unless noted otherwise. Note that a grouping may consist of only a subset of the data objects. For example, given tree $T^{\{f_1\}}$ in Figure 4.3, the following groupings are indicated by $T^{\{f_1\}}$:

- $G_1^{\{f_1\}} = \{\mathsf{S}(r_3^{f_1}), \mathsf{S}(r_5^{f_1})\} = \{\{s_2, s_3\}, \{s_4, s_5\}\}$.
- $G_2^{\{f_1\}}=\{\mathsf{S}(r_1^{f_1}), \mathsf{S}(r_2^{f_1})\}=\{\{s_1, s_2, s_3\}, \{s_4, s_5, s_6, s_7, s_8\}\}$

And the following groupings are not indicated by $T^{\{f_1\}}$:

- $\{\{s_1, s_6\}, \{s_4, s_7\}\}$.
- $\{\{s_1, s_2\}, \{s_3\}, \{s_5, s_7\}\}$.

The definition of parent-child groupings (Definition 2.3.5, Chapter 2) also applies. In this chapter, we call a child-grouping a finer grouping of its parent-grouping.

For example, given $G_3^{\{f_1\}} = \{\{s_1, s_2, s_3\}, \{s_4, s_5, s_7, s_8\}\}$, both

1. $G_4^{\{f_1\}} = \{\{s_1, s_2, s_3\}, \{s_4, s_5\}, \{s_7, s_8\}\}$
2. $G_5^{\{f_1\}} = \{\{s_1\}, \{s_2, s_3\}, \{s_4, s_5\}, \{s_7, s_8\}\}$

are its child-groupings. However, grouping

1. $G_6^{\{f_1\}} = \{\{s_2, s_3\}, \{s_4, s_5, s_7, s_8\}\}$
2. $G_7^{\{f_1\}} = \{\{s_1, s_2, s_3\}, \{s_6\}, \{s_4, s_5\}, \{s_7, s_8\}\}$

are not child-groupings of $P_3^{\{f_1\}}$.

Table 4.1: Notation Summary

H	data matrix, $H = S \times F$
S, s_i	the data object set, a data object
S',S'_i	a subset of data objects
F, f_i	the feature set, a feature
F',F'_i	a subset of features
$f_j(s_i)$	value of object s_i on feature f_j
$T^{F'}$	tree hierarchy on feature subset F'
$R^{F'}$, $r_i^{F'}$	tree node(s) of $T^{F'}$
$\mathsf{S}(r_i^{F'})$	object subset represented by node $r_i^{F'}$
$G_i^{F'}$	a grouping of objects implied by $T^{F'}$

4.4 Correlations and Problem Definition

In this section, we study the correlation among trees, groupings and features. We also provide a formal problem definition.

4.4.1 Correlation between a Grouping and a Feature

Given a grouping $G_i^{F'}$ indicated by $T^{F'}$ and a feature f_j,$f_j \in F - F'$, if $G_i^{F'}$ is correlated with f_j, objects in the same object subset of the partition tend to have similar f_j values while objects from different object subsets tend to have different f_j values. We use the data matrix in Figure 4.2 as an example. Given feature f_3 and two groupings indicated by trees $T^{\{f_1\}}$ and $T^{\{f_2\}}$ in Figure 4.3

- $G_8^{\{f_1\}} = \{\{s_2, s_3\}, \{s_4, s_5\}, \{s_7, s_8\}\}$
- $G_1^{\{f_2\}} = \{\{s_2, s_5\}, \{s_6, s_8\}, \{s_3, s_4\}\}$

Their feature values of f_3 are plotted in Figure 4.4, using different markers to represent objects in different subsets of a grouping. As we can see, grouping $G_8^{\{f_1\}}$ is more correlated with f_3 than $G_1^{\{f_2\}}$.

We use normalized RSS (residue squared sum of error) and BIC (Bayesian information criterion) to measure the correlation between a grouping and a feature.

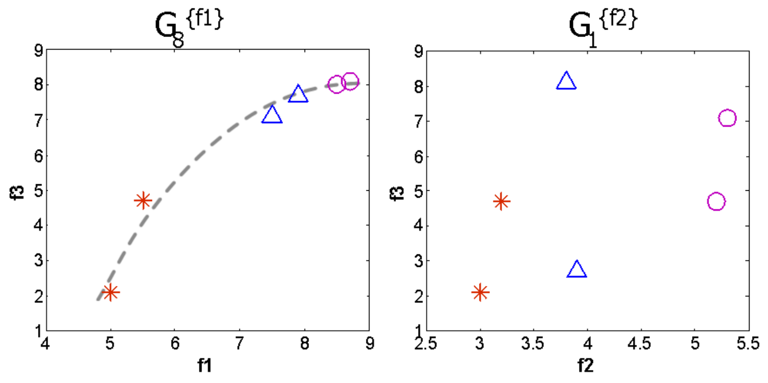


Figure 4.4: Correlation: feature f_3 and groupings $G_8^{\{f_1\}}$,$P_1^{\{f_2\}}$

Given a grouping $G_i^{F'} = \{S'_1, S'_2, ..., S'_u\}$, $S' = \bigcup_{l=1}^{u} S'_l$ and a feature f_j, we calculate the total variance SST, variance between object subsets SSB, and variance inside object subsets SSE as follows

$$M(S'_l) = \frac{1}{|S'_l|} \sum_{s_k \in S'_l} f_j(s_k), MM = \frac{1}{|S'|} \sum_{s_k \in S'} f_j(s_k) \tag{4.1}$$

$$SSB = \sum_{l=1}^{u} |S'_l|(M(S'_l) - MM)^2 \tag{4.2}$$

$$SSE = \sum_{l=1}^{u} \sum_{s_k \in S'_l} (f_j(s_k) - M(S'_l))^2 \tag{4.3}$$

$$SST = \sum_{s_k \in S'} (f_j(s_k) - MM)^2 = SSE + SSB \tag{4.4}$$

Then we get the normalized RSS

$$RSS(G_i^{F'}, f_j) = \frac{SSE}{SST} = \frac{SSE}{SSE + SSB} \tag{4.5}$$

For example, given the two partitions and feature in Figure 4.4, their RSS scores are

- $RSS(G_8^{\{f_1\}}, f_3) = \frac{3.57}{28.93} = 0.12$
- $RSS(G_1^{\{f_2\}}, f_3) = \frac{33.14}{34.14} = 0.97$

As we can see, a lower RSS value implies a stronger correlation.

Property 4.4.1. ***Monotonicity of RSS***: *Given two partitions $G_i^{F'}$ and $G_j^{F'}$, if $G_j^{F'}$ is a child (finer) partition of $G_i^{F'}$, for any feature f_u ($f_u \in F - F'$), we have $RSS(G_i^{F'}, f_u) \geq RSS(G_j^{F'}, f_u)$.*

Proof: Because $G_j^{F'}$ is a child (finer) partition of $G_i^{F'}$, $G_i^{F'}$ and $G_j^{F'}$ contain the same subset of objects. Therefore, we have

$$SST(G_i^{F'}, f_u) = SST(G_j^{F'}, f_u)$$

For each subset S'_{i_l} in $P_i^{F'}$

- If $\exists S'_{j_t} \in P_j^{F'}$, such that $S'_{i_l} = S'_{j_t}$, then

$$\sum_{s_k \in S'_{i_l}} (f_u(s_k) - M(S'_{i_l}))^2 = \sum_{s_k \in S'_{j_t}} (f_u(s_k) - M(S'_{j_t}))^2$$

- If S'_{i_l} is partitioned into $\{S'_{j_1}, S'_{j_2}, ..., S'_{j_w}\}$ in $G_j^{F'}$, then

$$\sum_{s_k \in S'_{i_l}} (f_u(s_k) - M(S'_{i_l}))^2 = \sum_{x=1}^{w} \sum_{s_k \in S'_{j_x}} (f_u(s_k) - M(S'_{j_x}))^2 + \sum_{x=1}^{w} |S'_{j_x}|(M(S'_{j_x}) - M(S'_{i_l}))^2$$

Thus,

$$\sum_{s_k \in S'_{i_l}} (f_u(s_k) - M(S'_{i_l}))^2 \geq \sum_{x=1}^{w} \sum_{s_k \in S'_{j_x}} (f_u(s_k) - M(S'_{j_x}))^2$$

Therefore, according to Equation 4.3, we have

$$SSE(G_i^{F'}, f_u) \geq SSE(G_j^{F'}, f_u)$$

Since $RSS = \frac{SSE}{SST}$, we get

$$RSS(G_i^{F'}, f_u) \geq RSS(G_j^{F'}, f_u)$$

$\square$

According to Property 4.4.1, if we measure correlation using RSS only, we will find that the finest groupings always have the lowest RSS score and hence the strongest correlation. To correct this bias of favoring finest groupings, we normalize the score by the number of subsets in the grouping since a finer grouping must contain a larger number of object subsets.

We utilize BIC (Bayesian information criterion) (Schwarz (1978)) to define the correlation between a grouping and a feature, taking into account RSS, the number of subsets and also the total number of objects in the grouping.

Definition 4.4.1. BIC Correlation between a Grouping and a Feature: Given grouping $G_i^{F'}$ and feature f_j ,$f_j \in F - F'$, the correlation between them is defined as

$$C(G_i^{F'}, f_j) = log(RSS(G_i^{F'}, f_j)) + u \cdot \frac{log(|S'|)}{|S'|} \tag{4.6}$$

in which, u is the number of object subsets in the grouping and $|S'|$ is total number of objects in the grouping.

A lower $C(G_i^{F'}, f_j)$ value indicates a stronger correlation between grouping $G_i^{F'}$ and feature f_j. For example, the correlation scores in Figure 4.4 are

- $C(G_8^{\{f_1\}}, f_3) = log(0.12) + 3 \cdot \frac{log(6)}{6} = -1.76$
- $C(G_1^{\{f_2\}}, f_3) = log(0.97) + 3 \cdot \frac{log(6)}{6} = 1.25$

The correlation score between $G_8^{\{f_1\}}$ and f_3 is much lower than the score between $G_1^{\{f_2\}}$ and f_3, which indicates that the correlation between $G_8^{\{f_1\}}$ and f_3 is much stronger.

The incorporation of u (the number of subsets in a grouping) into Equation 4.6 avoids the cases where finest groupings always have the lowest score. When two groupings have similar RSS scores, the BIC correlation favors the one containing smaller number of object subsets. Given two groupings which we used as examples before,

- $G_8^{\{f_1\}} = \{\{s_2, s_3\}, \{s_4, s_5\}, \{s_7, s_8\}\}$
- $G_6^{\{f_1\}} = \{\{s_2, s_3\}, \{s_4, s_5, s_7, s_8\}\}$

$G_8^{\{f_1\}}$ is a finer grouping of $G_6^{\{f_1\}}$. Their RSS values are similar

- $RSS(G_8^{\{f_1\}}, f_3) = 0.12$
- $RSS(G_6^{\{f_1\}}, f_3) = 0.13$

However, since $G_6^{\{f_1\}}$ contains fewer subsets, its BIC correlation score is lower than that of $G_8^{\{f_1\}}$.

- $C(G_8^{\{f_1\}}, f_3) = -1.76$
- $C(G_6^{\{f_1\}}, f_3) = -2.08$

The incorporation of $|S'|$ (the total number of objects (samples) in a partition) into Equation 4.6 is easy to see. A correlation supported by a large number of objects is always better than a correlation supported by a smaller number of objects.

As we discussed in Section 4.3, each tree hierarchy $T^{F'}$ can imply a set of groupings $\{G^{F'}\}$. Similar to TreeQA, we define the correlation between tree $T^{F'}$ and feature f_i, $f_i \in F - F'$ as the strongest correlation achieved by a grouping of $\{P^{F'}\}$ and feature f_i.

Definition 4.4.2. Correlation between a Tree and a Feature: Given tree $T^{F'}$ and feature f_i, $f_i \in F - F'$, the correlation between them is

$$C(T^{F'}, f_i) = min\{C(G_j^{F'}, f_i) | G_j^{F'} \in \{G^{F'}\}\} \tag{4.7}$$

4.4.2 Correlation Cluster

In TreeNL, a tree hierarchy $T^{F'}$ is constructed for each feature subset F'. We use $C(T^{F'}, f_i)$ to represent the correlation between feature subset F' and feature f_i.

Definition 4.4.3. Correlation between a Feature Subset and a Feature: Given a feature subset F' and a feature f_i, $f_i \in F - F'$, the correlation between them is

$$C(F', f_i) = C(T^{F'}, f_i) \tag{4.8}$$

The set of objects contained in the partition which achieves the strongest correlation with f_i is called the **support object (sample) subset of** $C(F', f_i)$ and is denoted by $D(F', f_i)$.

For example, given a feature subset $\{f_1\}$ and a feature f_3 in Figure 4.2, TreeNL constructs a tree hierarchy on feature subset $\{f_1\}$ which is plotted in Figure 4.3. After examining all groupings implied by $T^{\{f_1\}}$, pgrouping $\{\{s_2, s_3\}, \{s_4, s_5, s_7, s_8\}\}$ is found to achieve the lowest BIC correlation score -2.08. Therefore, $C(\{f_1\}, f_3) = -2.08$ and $D(\{f_1\}, f_3) = \{s_2, s_3, s_4, s_5, s_7, s_8\}$.

If we want to know the correlation between features in a given feature subset F', (how closely are the features in F' related with each other?), we can select any feature f_i from F' and calculate the correlation between f_i and the rest of the features $(F' - \{f_i\})$.

Definition 4.4.4. Correlation of a Feature Subset: Given a feature subset F', the correlation of F' is

$$C(F') = min\{C(F' - \{f_i\}, f_i) | f_i \in F'\} \tag{4.9}$$

And the support object subset corresponding to the strongest correlation is considered as the **support object (sample) subset of** $C(F')$, which is denoted by $D(F')$.

Definition 4.4.5. Correlation Cluster: Given a feature subset F', its support object (sample) subset $D(F')$ forms a correlation cluster. The correlation score of F', $C(F')$, indicates the quality of the correlation cluster.

4.4.3 Problem Definition

Given a data matrix $H = S \times F$, find the top-K correlation clusters (i.e., correlation feature subsets and their corresponding support object subsets).

4.5 TreeNL Algorithm

We present the TreeNL algorithm in this section. Given a data matrix $H = S \times F$, TreeNL outputs the K most significant correlation clusters (correlated feature subset and the corresponding support object subset) allowing arbitrary overlapping between clusters.

We set 3 input parameters to the TreeNL algorithm.

- K: the number of most significant clusters to output.
- $fset_{max}$: the maximum size of feature subsets examined by TreeNL.
- g_{min}: the minimum size of an object subset represented by a node in the tree hierarchy.

TreeNL computes the BIC correlation score for each enumerated feature subset. Most of the subsets have no or weak correlations. The top-K output provides a way to identify significant correlation clusters. Alternatively, one may set a threshold on the BIC correlation score which can be easily incorporated in TreeNL.

For datasets containing hundreds of features, an exhaustive search of all subsets of features is time-consuming and unnecessary. The correlation of a large feature subset

```
Main Routine
Input:
  • Data matrix $H = S \times F$.
  • $K$, $fset_{max}$, $g_{min}$.
Output: Top-$K$ correlation clusters.
Method:
  1. impose an order on the features, s.t.,
     $F = \{f_1, f_2, ..., f_m\}$.
  2. Initialize the top-$K$ cluster queue, $Q_K = \emptyset$.
  3. $curlist = \emptyset$,$remlist = F$.
  4. EnumerateFeature( $curlist$, $remlist$ ).
  5. report the $K$ most significant correlation clusters.

Subroutine: EnumerateFeature( $curlist$, $remlist$ )
Method:
  1. for each remaining feature $f_i$ in $remlist$
  2.    $flist$=$curlist \cup \{f_i\}$.
  3.    for each feature $f_j \in flist$
  4.       $T^{flist-\{f_j\}}$=TreeConstruct($flist - \{f_j\}$).
  5.       sort the tree nodes in breadth-first order
           $R^{flist-\{f_j\}} = \{r_0, r_1, r_2, ..., r_w\}$.
  6.       $Pcurlist = \emptyset$, $Premlist = R^{flist-\{f_j\}}$.
  7.       $[C(T^{flist-\{f_j\}}, f_j)$, $D(T^{flist-\{f_j\}}, f_j)]$
           = EnumeratePartition($Pcurlist, Premlist, f_j$).
  8.       $C(flist - \{f_j\}, f_j) = C(T^{flist-\{f_j\}}, f_j)$.
  9.    $C(flist) = min\{C(flist - \{f_j\}, f_j)\}$.
 10.    store $D(flist)$ accordingly.
 11.    update the top-$K$ cluster queue $Q_K$.
 12.    remove $f_i$ from $remlist$.
 13.    if $|flist| < fset_{max}$
 14.       EnumerateFeature($flist$, $remlist$).

Subroutine: EnumeratePartition($Pcurlist$, $Premlist$, $f$)
Method:
  1. for each remaining tree nodes $r_i$ in $Premlist$
  2.    $Plist$=$Pcurlist \cup \{r_i\}$.
  3.    generate grouping $G = \{\mathtt{S}(r_j) | r_j \in Plist\}$.
  4.    if $G$ is not a valid grouping (Definition ??)
  5.       return the best correlation found so far.
  6.    calculate $C(G, f)$.
  7.    compare with the current best correlation, and update.
  8.    remove tree node $r_i$ from $Premlist$.
  9.    EnumeratePartition($Plist$, $Premlist$, $f$).
 10. return the best correlation so far.
```

Figure 4.5: The TreeNL Algorithm

may often be explained by its smaller subsets. $fset_{max}$ defines the maximum size of the feature subsets examined by TreeNL. We usually set this parameter to 4 or 5.

g_{min} is used to ensure that each object (sample) subset contains sufficient objects so that the correlation calculated between groupings and features are statistically significant. It also controls the size of the tree hierarchies to make enumeration efficient. We usually set this number to be at least 1% of the total number of data objects.

Given a data matrix, TreeNL enumerates all subsets of features up to size $fset_{max}$. For each feature subset F', TreeNL selects a feature f_j from F' one at a time. TreeNL constructs a tree hierarchy $T^{F'-\{f_j\}}$ to organize all data objects according to their similarities (e.g., Euclidian distance in feature subspace $F' - \{f_j\}$). Each node in tree $T^{F'-\{f_j\}}$ represents an object subset containing at least g_{min} objects. TreeNL then enumerates all groupings indicated by the tree hierarchy and calculate the BIC correlation $C(P^{F'-\{f_j\}}, f_j)$. According to Definitions 4.4.2 and 4.4.3, the strongest correlation achieved by the groupings is considered as the correlation between $F' - \{f_j\}$ and f_j. After selecting every feature from F', TreeNL gets the correlation of the subset F' and its corresponding support object subset (cluster). In the end, TreeNL outputs the K most significant correlation clusters (or reports the correlation score of each cluster).

Figure 4.5 shows the pseudocode of the basic implementation of TreeNL. We will discuss the improvements later in this section.

In the **Main Routine**, an order is imposed on the features in F in Step 1 so that TreeNL can systematically enumerate the feature subset. The **EnumerateFeature** routine is called in Step 4.

In **EnumerateFeature**, a new feature subset $flist$ is generated in Step 2. From Step 3 to Step 7, in each loop, a feature $f_j \in flist$ is removed. The tree hierarchy is constructed on $flist - \{f_j\}$ in Step 4. The tree nodes of $T^{flist-\{f_j\}}$ are sorted in a breadth-first order in Step 5 so that all partitions implied by $T^{flist-\{f_j\}}$ can be generated by enumerating the tree node subsets systematically in **EnumeratePartition**. Routine **EnumeratePartition** is called in Step 7 to get the correlation between tree $T^{flist-\{f_j\}}$ and feature f_j. In Step 9, the correlation of feature subset $flist$ is calculated according to Definition 4.4.4. The corresponding support object subset is stored in Step 10 and the top-K correlation cluster Q_K is updated in Step 11. If the number of features in $flist$ is less than $fset_{max}$, **EnumerateFeature** is recursively called in Step 14.

In **EnumeratePartition**, a new tree node subset $Plist$ is generated in Step 2 in each loop. The corresponding grouping $G = \{\mathtt{S}(r_j)|r_j \in Plist\}$ is generated and validated in Steps 3 and 4. If G is not a valid grouping, (e.g., there is overlapping between the object subsets represented by the nodes in $Plist$) **EnumeratePartition** returns directly in Step 5 with the best correlation found so far. Otherwise, the correlation between the grouping and the feature is calculated in Step 6. In Step 7, the current best correlation is compared with the result from Step 6 and is updated accordingly. **EnumeratePartition** is recursively called in Step 9. At the end of **EnumeratePartition**, the best correlation is returned.

4.5.1 Tree Hierarchy Construction

Both top-down and bottom-up hierarchical clustering approaches are able to construct a tree hierarchy. TreeNL requires that each node in the tree must represent at least g_{min} objects. Usually we set g_{min} to be at least 1% of the total number of objects. A top-down approach is more efficient than a bottom-up approach, because a bottom-up approach constructs a tree from leaf nodes representing single objects.

TreeNL utilizes a k-means based top-down hierarchical clustering approach to construct tree hierarchy. Starting from the root node which represents the entire set of data objects, routine **TreeConstruct** continuously partitions the subset of data object at each node into two disjoint subsets. The two new subsets are then represented by the left and right child-nodes of the current node. The partition continues until the number of objects in the subset is less than g_{min}. **TreeConstruct** uses 2-means clustering to partition each subset of objects into two subsets. Note that any other top-down hierarchical clustering approaches can also be used in TreeNL to construct tree hierarchy.

The pseudocode code of routine **TreeConstruct** is in Figure 4.6. In the main routine, the root node of the tree $r_0^{F'}$ is initialized in Steps 1 and 2. Subroutine **Partition** is called in Step 3. In **Partition**, two random seeds are selected from the input object

Routine: TreeConstruct
Input:

- Data matrix $H = S \times F$.
- Feature subset, F'
- g_{min}.

Output: Tree hierarchy $T^{F'}$ on feature subset F'.
Method:

```
1. generate the root node $r_0^{F'}$ of tree
2. let $\mathtt{S}(r_0^{F'}) = S$.
3. **Partition**($\mathtt{S}(r_0^{F'})$, $r_0^{F'}$).
4. return tree $T^{F'}$.
```

Subroutine: Partition($\mathtt{S}(r_i^{F'})$, $r_i^{F'}$)
Method:

```
1. randomly select two objects from $\mathtt{S}(r_i^{F'})$ as the initial centroids.
2. while at least one centroid changes
3.    for each object in $\mathtt{S}(r_i^{F'})$
4.       get the Euclidian distance in $F'$ to both centroids.
5.       assign object to the closer centroid.
6.    re-calculate the centroids for both clusters.
7. let the two clusters be $S'_1$ and $S'_2$,
   $S'_1 \cap S'_2 = \emptyset$, $S'_1 \cup S'_2 = \mathtt{S}(r_i^{F'})$.
8. if $|S'_1| < g_{min}$ or $|S'_2| < g_{min}$
9.    stop partition. return.
10. initialize the left and right child-nodes of $r_i^{F'}$. let them be
    $r_u^{F'}$ and $r_v^{F'}$.
11. $\mathtt{S}(r_u^{F'}) = S'_1$, $\mathtt{S}(r_v^{F'}) = S'_2$.
12. **Partition**($\mathtt{S}(r_u^{F'})$, $r_u^{F'}$).
13. **Partition**($\mathtt{S}(r_v^{F'})$, $r_v^{F'}$).
```

Figure 4.6: Routine **TreeConstruct**

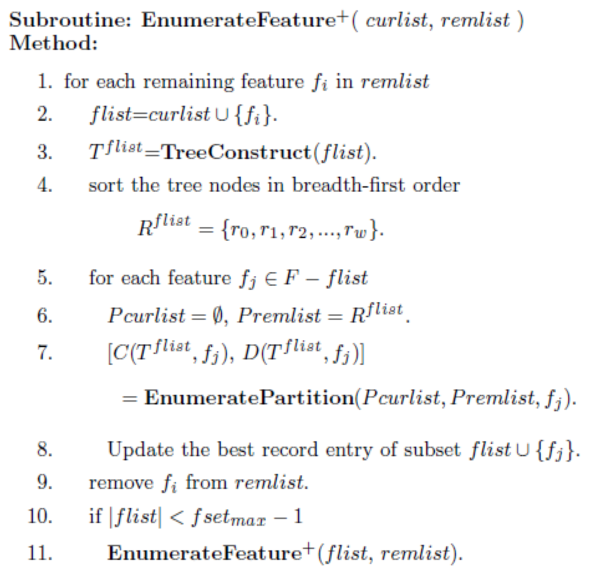
subset $\mathtt{S}(r_i^{F'})$. From Step 2 to Step 6, k-means clustering is used to partition $\mathtt{S}(r_i^{F'})$ into two disjoint subsets S'_1 and S'_2. If the size of either subset is less than g_{min}, the partition stops and the routine returns directly in Step 9. Otherwise, the left and right child-nodes of $r_i^{F'}$ are initialized in Step 10. Each child-node represents one of the subsets. And routine **TreeConstruct** is recursively called in Steps 12 and 13 for the left and right child-nodes respectively.

4.5.2 A Faster Enumeration

In routine **EnumerateFeature** (Figure 4.5), given a feature subset $flist$, a tree hierarchy is constructed for each $flist - \{f_j\}$ subset. We found that the same tree is actually repeatedly constructed in different $flist$ subsets. For example, if two feature subsets

$\{s_1, s_2, s_3\}$ and $\{s_1, s_2, s_4\}$ are both enumerated, tree $T^{\{s_1,s_2\}}$ are constructed in both of them.

In order to avoid this repeated tree construction, we re-design the routine. During the enumeration of features, given a feature subset $flist$, we construct tree T^{flist} and calculate the correlation between T^{flist} and each f_j, $f_j \in F - flist$. By doing so, we ensure that for any feature subset F', tree $T^{F'}$ is constructed only once. However, this method causes another problem. The correlation of a feature subset F' depends on all $C(F' - \{f_i\}, f_i)$, $f_i \in F'$. With the new method, we need to keep an entry for each subset F' to record the best $C(F' - \{f_i\}, f_i)$ achieved so far. Then for any feature subset F', its correlation $C(F')$ can be known after the entire enumeration. In our implementation, we use a hash list to organize the best record entry for each subset F'.



```
Subroutine: EnumerateFeature+( curlist, remlist )
Method:
 1. for each remaining feature $f_i$ in remlist
 2.     $flist = curlist \cup \{f_i\}$.
 3.     $T^{flist}$=TreeConstruct($flist$).
 4.     sort the tree nodes in breadth-first order
            $R^{flist} = \{r_0, r_1, r_2, ..., r_w\}$.
 5.     for each feature $f_j \in F - flist$
 6.         $Pcurlist = \emptyset$, $Premlist = R^{flist}$.
 7.         $[C(T^{flist}, f_j), D(T^{flist}, f_j)]$
                = EnumeratePartition($Pcurlist, Premlist, f_j$).
 8.         Update the best record entry of subset $flist \cup \{f_j\}$.
 9.     remove $f_i$ from remlist.
10.     if $|flist| < fset_{max} - 1$
11.         EnumerateFeature+($flist$, remlist).
```

Figure 4.7: Re-designed Routine **EnumerateFeature**$^+$

The pseudocode code of the new **EnumerateFeature**$^+$ routine is in Figure 4.7. From Step 2 to Step 4, tree T^{flist} is constructed for subset $flist$. From Step 5 to Step 8, the correlation between T^{flist} and each feature not included in $flist$ is calculated. The corresponding best record entries are also updated accordingly. Note that in Step 10, the stop condition changes a little bit because the feature subset $flist \cup \{f_j\}$ (instead of $flist$) is checked in Steps 7 and 8.

4.5.3 Other Improvements

On each node $r_j^{F'}$ of tree $T^{F'}$, we store the corresponding object subset $\mathtt{S}(r_j^{F'})$. This object subset is used in two places during the procedure.

- Overlapping check: In Steps 3 and 4 of routine **EnumeratePartition** (Figure 4.5), these object subsets are used to generate a grouping $G^{F'}$ and are used in its validation to check if there is any overlapping between these object subsets.
- Correlation calculation: In Step 6 of routine **EnumeratePartition** (Figure 4.5), these object subsets are used to calculate the correlation score between a grouping and a feature.

In fact, we don't need these $\mathtt{S}(r_j^{F'})$ subsets to be physically stored at each node except for the leaf nodes.

Given two nodes in tree $T^{F'}$, their corresponding object subsets overlap if and only if one of the nodes is a child-node of the other one. Therefore, instead of validating grouping $G^{F'}$ in Step 4 of **EnumeratePartition**, we can check the parent-child relationship between r_i and each node in $Pcurlist$ in Step 2 of **EnumeratePartition** and add r_i to $Pcurlist$ only when there is no such relationship between them. The relationship checking can be efficiently done using the tree hierarchy with some additional pointers between nodes.

Given a feature f_i, for the correlation calculation, we only need to store the number of objects in $\mathtt{S}(r_j^{F'})$ and the following two values for each node $r_j^{F'}$

$$SQ(\mathtt{S}(r_j^{F'})) = \sum_{s_t \in \mathtt{S}(r_j^{F'})} f_i(s_t)^2 \tag{4.10}$$

$$SM(\mathtt{S}(r_j^{F'})) = \sum_{s_t \in \mathtt{S}(r_j^{F'})} f_i(s_t) \tag{4.11}$$

To calculate the RSS of a grouping $G^{F'} = \{\mathtt{S}(r_j^{F'})|j = 1...w\}$ and the feature f_i, for each included subset (node), we calculate

$$SSE_j = SQ(\mathtt{S}(r_j^{F'})) - SM(\mathtt{S}(r_j^{F'}))^2/|\mathtt{S}(r_j^{F'})| \tag{4.12}$$

$$SSB_j = SM(\mathtt{S}(r_j^{F'}))^2/|\mathtt{S}(r_j^{F'})| \tag{4.13}$$

Then we combine them together and get

$$N = \sum_{j=1}^{w} |\mathtt{S}(r_j^{F'})|, MM = \frac{1}{N}\sum_{j=1}^{w} SM(\mathtt{S}(r_j^{F'})) \tag{4.14}$$

$$SSB = \sum_{j=1}^{w} SSB_j - N \cdot MM^2, SSE = \sum_{j=1}^{w} SSE_i \tag{4.15}$$

$$RSS = \frac{SSE}{SSE + SSB} \tag{4.16}$$

The correlation $C(G^{F'}, f_i)$ can be easily computed after we get the RSS.

Therefore, except for the leaf nodes, we do not need to physically store $\mathtt{S}(r_j^{F'})$ for the nodes. And the operations (overlapping check and correlation calculation) can be done more efficiently.

4.6 Experiments

In this section, we present experiment results on both synthetic and real data to demonstrate the efficiency and effectiveness of our TreeNL algorithm. The experiments are performed on a 2.4 GHz PC with 1G memory running WindowsXP system.

Datasets

- Synthetic dataset: We generated a set of synthetic data with different sizes. Both linear and nonlinear correlation clusters are embedded in the data matrix.
- NBA statistics dataset: The data was downloaded from the ESPN web page[3]. Each data object is an NBA player. Each player is characterized by 28 features (statistics), such as ”number of blocks”, ”number of points”, etc...
- Mouse gene expression dataset: The dataset was provided by the School of Public Health at UNC. The dataset contains the expression values of 101 genes in 42 mouse samples.

Algorithms

- CURLER (Tung et al. (2005)): a nonlinear correlation clustering algorithm which outputs a strict partition of data objects.
- CARE (Zhang et al. (2008)): a linear correlation clustering algorithm which allows arbitrary overlapping between clusters.
- TreeNL: basic implementation of TreeNL.
- TreeNL$^+$: TreeNL with re-designed **EnumerateFeature**$^+$ routine and other improvements discussed in Section 4.5. Note that TreeNL$^+$ outputs the same clustering results as TreeNL, except that TreeNL$^+$ is faster.

Parameter Setting

In the experiments, we use the default parameter settings of CURLER and CARE according to (Tung et al. (2005); Zhang et al. (2008)), except that we vary the number

[3]http://sports.espn.go.com/nba/teams/stats?team=Bos year=2007 season=2

of micro-clusters which are generated in CURLER to get the best output of CURLER. In this section, we use term 'CURLER$_t$' in the captions of the corresponding figures to denote that CURLER generated t micro-clusters on the dataset.

For TreeNL, we vary g_{min} and $fset_{max}$ in the following ranges in different experiments,

- g_{min}: vary from 2% to 5% of the total number of objects.
- $fset_{max}$: vary from 2 to 5.

Parameter K is user-defined and is used only when TreeNL outputs the most significant clusters.

4.6.1 Synthetic Data: Effectiveness

We generated three synthetic data for this set of experiments.

- Syndata1: A 400×6 data matrix. Two nonlinear helix correlation clusters are embedded to resemble the synthetic data used in CURLER (Tung et al. (2005)). The clusters and nonlinear dependencies are listed in Table 4.2.
- Syndata2: A 100×100 data matrix. Two linear correlation clusters are embedded to resemble the synthetic data used in CARE (Zhang et al. (2008)). The clusters and linear dependencies are listed in Table 4.2.
- Syndata3: A 300×7 data matrix. This dataset has been used as the example in Figure 4.1. Two linear correlation clusters and one nonlinear correlation cluster are embedded. The three clusters overlap with each other on both features and objects. The clusters and linear/nonlinear dependencies are listed in Table 4.2.

Table 4.2: Correlation Clusters Embedded in Syndata1,2 and 3

Syndata1		
Cluster	Point Subset	Feature Dependency
1	$\{s_1, ..., s_{200}\}$	$f_1 = 2 \cdot t, f_2 = 1.2 \cdot sin(t)$, $f_3 = 1.2 \cdot cos(t)$, $t \in [0, 6\pi]$
2	$\{s_{201}, ..., s_{400}\}$	$f_4 = u, f_5 = 2 \cdot sin(u)$, $f_6 = 2 \cdot cos(u)$, $u \in [0, 6\pi]$
Syndata2		
Cluster	Point Subset	Feature Dependency
1	$\{s_1, ...s_{60}\}$	$f_{10} = f_8 + f_9$
2	$\{s_{41}, ...s_{100}\}$	$f_{50} = f_{48} + f_{49},$
Syndata3		
Cluster	Point Subset	Feature Dependency
1	$\{s_1, ...s_{100}\}$	$f_2 = f_1 + 0.5 \cdot f_3$
2	$\{s_{61}, ...s_{160}\}$	$f_7 = f_5 + f_6,$
3	$\{s_{120}, ...s_{219}\}$	$f_4 = 2 \cdot f_3 \cdot f_5,$

Syndata1

Since both embedded correlations are nonlinear, CARE found no clusters on Syndata1.

The output of CURLER is a visualization of the clusters called NNCO plot (Tung et al. (2005)). The NNCO plot on Syndata1 is shown in Figure 4.8(a). The x-axis denotes the micro-clusters which are ordered according to the cluster merging procedure of CURLER. The y-axis denotes the co-sharing level of micro-clusters. In general, a cluster will be represented by a hill shape in the NNCO plot. The bars below the co-sharing plot represent the orientations of the micro-clusters. For micro-clusters in the same cluster, their orientations are similar and therefore, a block (or a pattern) in the corresponding bars can be observed graphically. Interested readers may refer to (Tung et al. (2005)) for details. Note that in the caption of Figure 4.8, term 'CURLER$_{400}$' denotes that CURLER generated 400 micro-clusters on this dataset.

We can observe two hills in Figure 4.8(a), one from micro-clusters 1 to 200 and the other from micro-clusters 201 to 400. These two hills clearly indicate the existence of the two embedded nonlinear (helix) clusters.

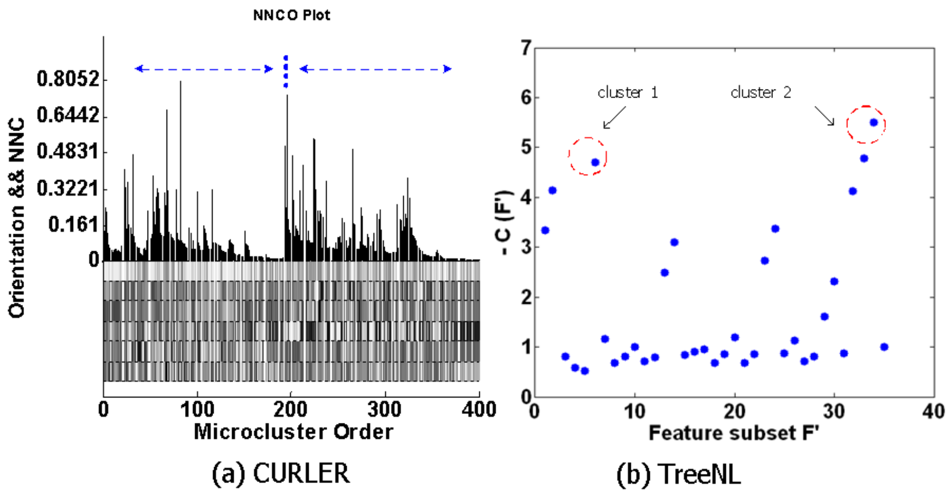


Figure 4.8: Outputs of CURLER$_{400}$ and TreeNL on Syndata1

For TreeNL, we set $g_{min} = 10$ and $fset_{max} = 3$. And instead of outputting the top-K clusters (feature subsets), we output the correlation score for all enumerated clusters (feature subsets) for fair comparison.

Figure 4.8(b) plots the output of TreeNL on Syndata1, that is, the $-C(F')$ score of each enumerated feature subset. The x-axis denotes the feature subsets in lexicographic order. The y-axis denotes the $-C(F')$ score. Note that a higher point in the figure indicates a stronger correlation.

We can observe two peak points in Figure 4.8(b). The left peak represents feature subset $\{f_1, f_2, f_3\}$ which corresponds to cluster 1, and the right peak represents feature subset $\{f_4, f_5, f_6\}$ which corresponds to cluster 2. For each peak, there are two other points to the left which also indicate strong correlation. These points represent feature subsets $\{f_1, f_2\}$, $\{f_1, f_3\}$, $\{f_4, f_5\}$ and $\{f_4, f_6\}$ respectively. According to the dependency functions in Table 4.2, the correlations between these features are obvious.

On Syndata1, both CURLER and TreeNL found the embedded correlations while CARE failed.

Syndata2

CARE successfully detects the two embedded linear correlations and provides the quantitative information of the dependencies,

- $f_8 + 1.02 \cdot f_9 - 0.98 \cdot f_{10} = 0$
- $f_{48} + 0.99 \cdot f_{49} - 0.97 \cdot f_{50} = 0$

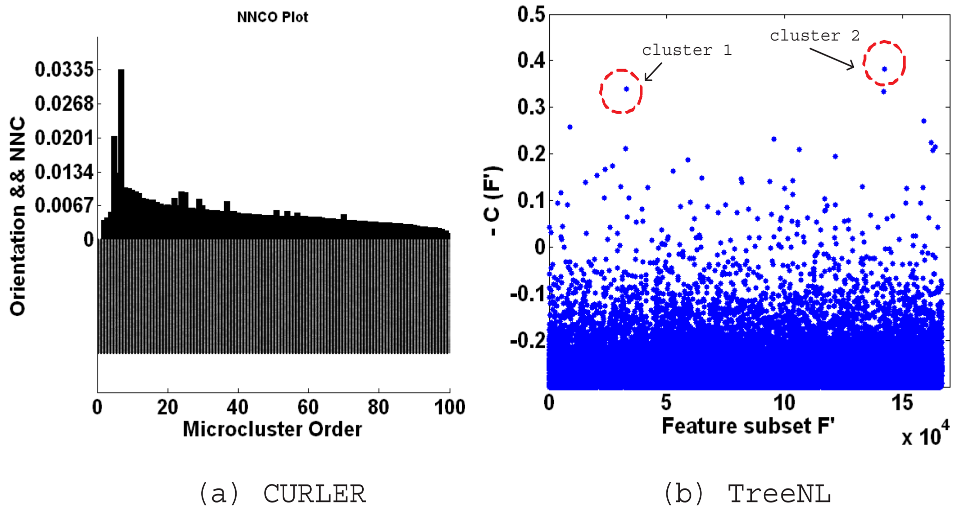


Figure 4.9: Outputs of CURLER$_{100}$ and TreeNL on Syndata2

The NNCO plot of CURLER is shown in Figure 4.9(a). There are no obvious hills which can indicate the two embedded clusters. Both the high intrinsic dimensionality of the data (94% of the features are random noise) and the overlapping of data objects between the two clusters prevent CURLER from finding the clusters.

For TreeNL, we still use $g_{min} = 10$ and $fset_{max} = 3$. Figure 4.9(b) plots the output of TreeNL on Syndata2. The two top points in Figure 4.9(b) represent feature subsets $\{f_8, f_9, f_{10}\}$ and $\{f_{48}, f_{49}, f_{50}\}$ which correspond to the two embedded clusters.

On Syndata2, both CARE and TreeNL found the embedded correlations while CURLER failed.

Syndata3

CARE didn't find the nonlinear correlation cluster. Since all three correlated feature subsets are supported by a minority of data objects (30%), CARE found only one linear correlation

- $f_1 - 0.97 \cdot f_2 + 0.51 \cdot f_3 = 0$

If we relax the parameters of CARE, e.g., lowering the minimum support threshold, the second correlation will then be found together with many other weakly correlated and spurious clusters.

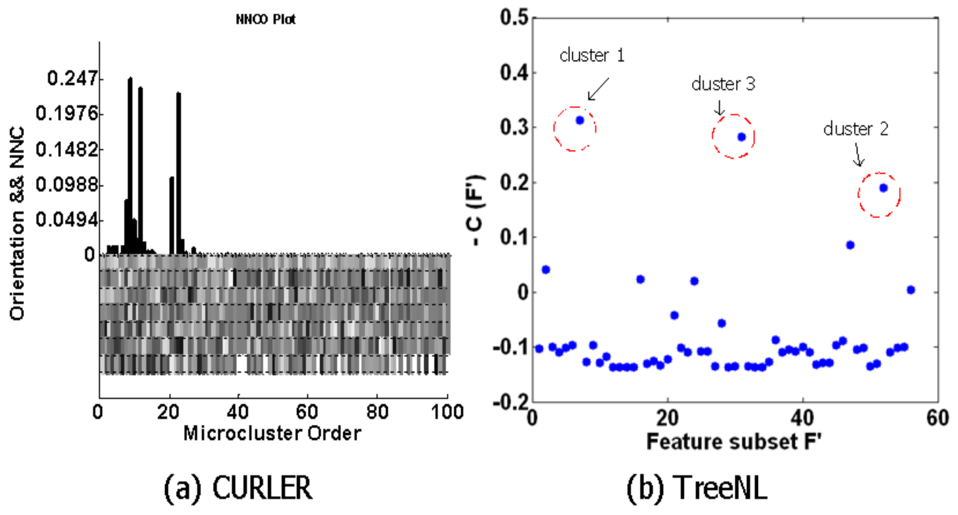


Figure 4.10: Outputs of CURLER_{100} and TreeNL on Syndata3

The NNCO plot of CURLER is shown in Figure 4.10(a). There are two small hills plotted in Figure 4.10(a). The micro-clusters corresponding to these hills contain parts of the objects in clusters 1 and 3. Because of the substantial overlapping between the embedded clusters, CURLER didn't find cluster 2 (see Table 4.2) and only found parts of clusters 1 and 3.

For TreeNL, we use the same setting, $g_{min} = 10$ and $fset_{max} = 3$. Figure 4.10(b) plots the output of TreeNL on Syndata3. The top three points represent feature subsets

$\{f_1, f_2, f_3\}$, $\{f_3, f_4, f_5\}$ and $\{f_5, f_6, f_7\}$ which correspond to the three embedded clusters. The data objects in each cluster returned by TreeNL are plotted in Figure 4.11. Compared with the embedded clusters shown in Figure 4.1, we can see that TreeNL can discover both linear and nonlinear correlations very accurately.

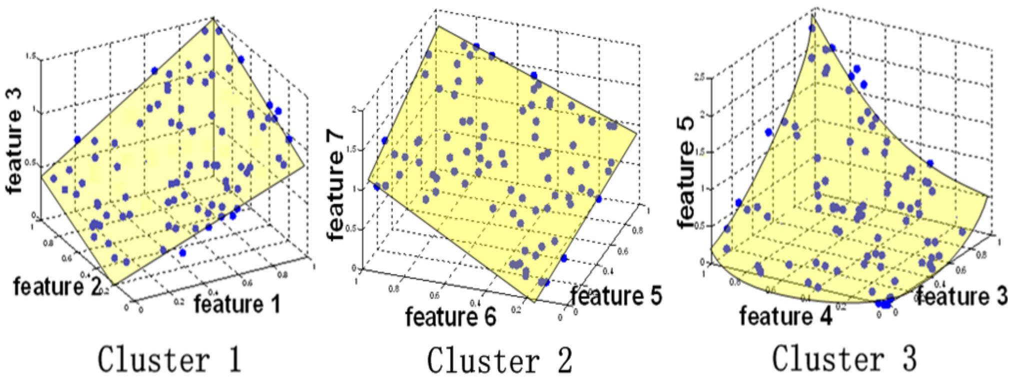


Figure 4.11: Clusters of objects found by TreeNL on Syndata3

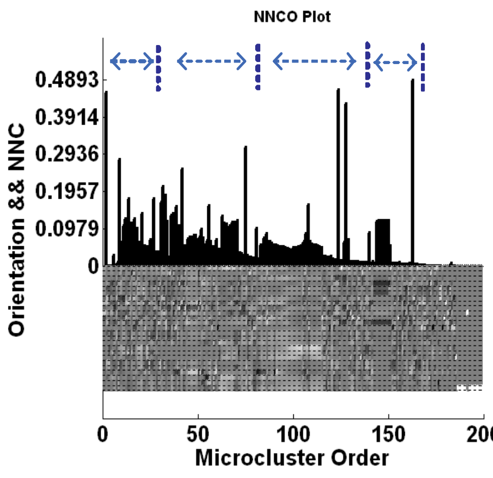
On Syndata3, TreeNL found all three embedded clusters while CARE and CURLER only found some of them.

4.6.2 NBA Data: Effectiveness

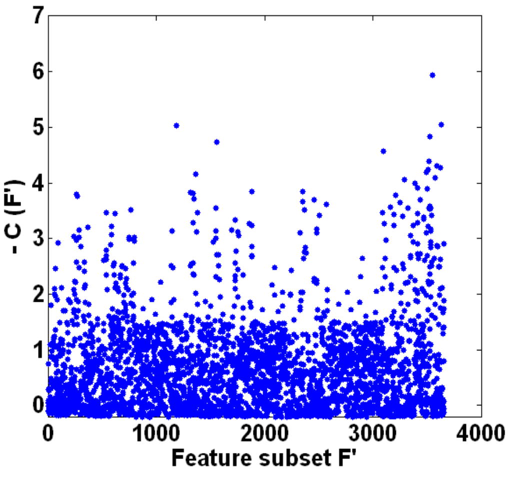
In this section, we present the experiment results on a real data: NBA dataset. The dataset was collected for 200 players from the game season in 2007. For each player in the dataset, 28 statistics are used as features. The dataset contains both linear and nonlinear correlations.

For TreeNL, we set its parameters as $g_{min} = 10$ and $fset_{max} = 3$. The score for each enumerated feature subset (cluster) is plotted in Figure 4.12(b). The ***top-5*** **correlation clusters** in Figure 4.12(b) correspond to the following 5 correlated feature subsets:

1. 'number of defense rebounds', 'number of offense rebounds', 'total number of rebounds'.



(a) CURLER



(b) TreeNL

Figure 4.12: Outputs of CURLER_{200} and TreeNL on the NBA dataset

2. '3-point field goal made'(3PM), '3-point attempted'(3PA), '3-point made percentage' (3PM/3PA).

3. 'free throw made'(FTM), 'free throw attempted'(FTA), 'free throw percentage'(FTM/FTA).

4. 'game played', '2-point made percentage', 'free throw percentage'.

5. 'number of assist'(AST), 'number of turnover'(TO), 'AST/TO'.

The correlations of feature subsets 1, 2, 3 and 5 are obvious. The correlation of subset 4 implies that a subset of players (a cluster) who are good at 2-point shooting and free throw get more chance to play games. Due to space limitation, we only plot the correlation clusters corresponding to subsets 1 and 2 in Figure 4.13.

The NNCO plot of CURLER is shown in Figure 4.12(a). From the figure, we can recognize 4 clusters. According to the bars, the two on the right correspond to feature subsets 1 and 3 respectively; while the two clusters on the left are hard to interpret. CURLER didn't find other correlated clusters because of the substantial overlapping of objects between the clusters. The statistics of a player can have different dependencies in different feature subsets.

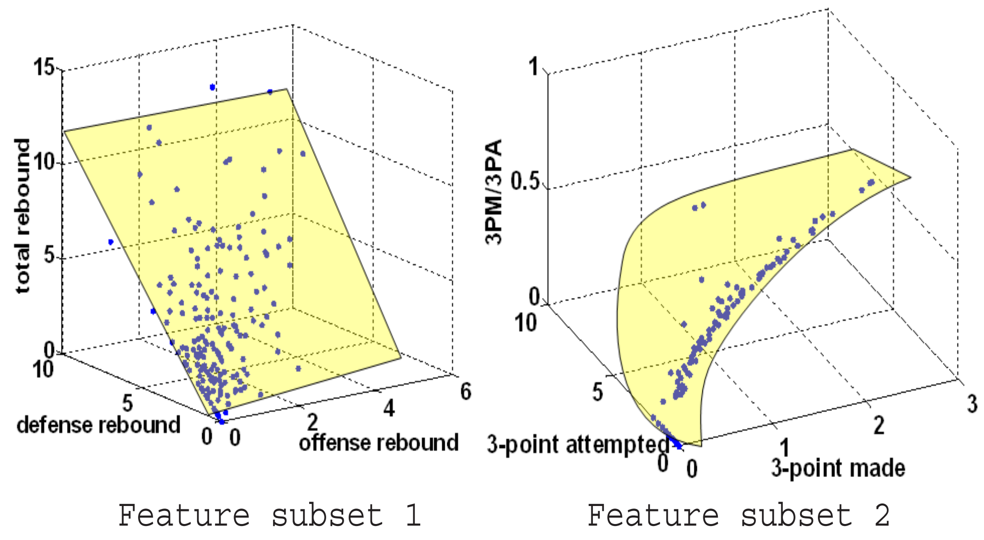


Figure 4.13: Correlation clusters corresponding to correlated feature subsets 1 and 2

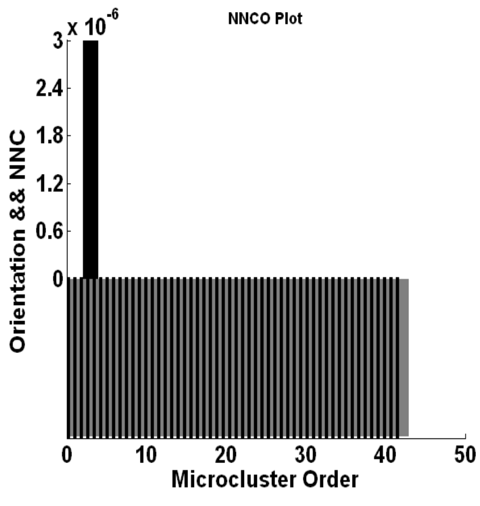
Out of the 5 correlation clusters found by TreeNL, CARE found two of them (subsets 1 and 4) which are linear correlations. CARE also found other linear correlation clusters in the NBA data. Those correlation clusters were also found by TreeNL (though not in the top-5) with relatively high $-C(F')$ scores.

4.6.3 Mouse Gene Expression Data: Effectiveness

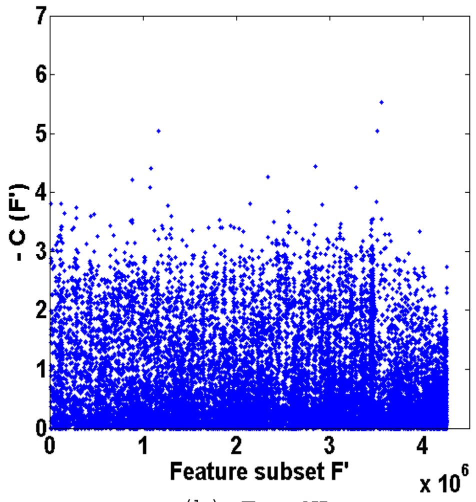
In this section, we apply the algorithms on the mouse gene expression data. This data contain 101 gene expression values of 42 mouse samples. Correlation clusters found in this dataset overlap with each other for both genes and samples, that is, a gene can participate in different correlated gene subsets, and a mouse sample can also occur in different correlation clusters in different gene subsets. This dataset was used in CARE (Zhang et al. (2008)) and several linearly correlated gene subsets were reported.

We set TreeNL's parameters as $g_{min} = 4$ and $fset_{max} = 4$. The scores of all enumerated feature (gene) subsets (clusters) are plotted in Figure 4.14(b).

Among the top-5 correlated subsets in Figure 4.14(b), the following two have been reported by CARE,



(a) CURLER



(b) TreeNL

Figure 4.14: Outputs of CURLER$_{42}$ and TreeNL on the Mouse Gene Expression dataset

- { Nrg4, Myh7, Hist1h2bk, Arntl }

- { Oazin, Ctse, Mgst3 }

The other three correlated gene subsets (clusters) were not found by CARE because their corresponding correlation clusters contain less than 20 mouse samples. Even though we can relax the parameters of CARE, these three clusters will be overwhelmed by many other weakly correlated and spurious clusters. We show the three subsets in Table 4.3 with the gene IDs and GO (gene ontology) annotations. As we can see in the table, genes in each subset have consistent annotations.

We also plot the corresponding correlation clusters (support samples subsets) of gene subsets 1, 2 and 3 (in Table 4.3) in Figure 4.15.

The NNCO plot of CURLER is shown in Figure 4.14(a). Since the clusters in this dataset have substantial overlapping, it is hard to interpret any clusters from the NNCO plot.

Table 4.3: Correlated Gene Subsets

Subset	Gene IDs	GO annotations
1	Prc1	cytokinesis
	Lcp2	cytokine secretion
	G1p2	response to virus
	lfi27	response to virus
2	Ldb3	intracellular part
	Sec61g	intracellular part
	Exosc4	intracellular part
	BC048403	N/A
3	Ptk6	membrane
	Gucy2g	integral to membrane
	Clec2g	integral to membrane
	H2-Q2	integral to membrane

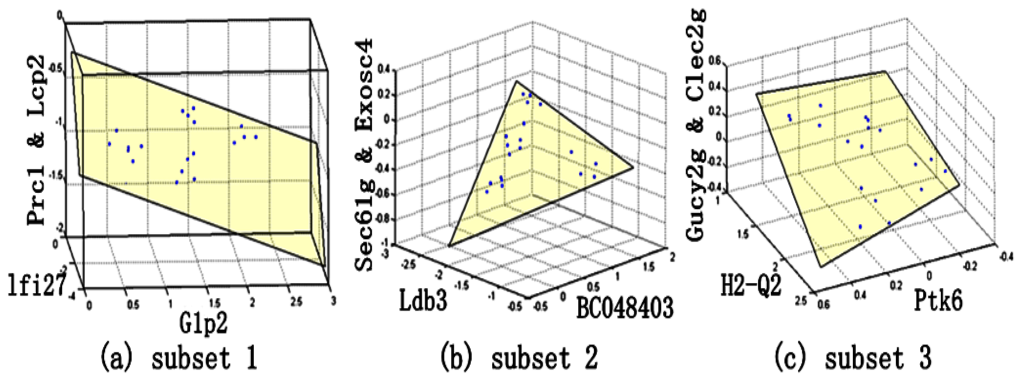


Figure 4.15: Correlation clusters of samples correspond to gene subsets in Table 4.3

4.6.4 Synthetic Data: Efficiency

We generated a set of synthetic data of different sizes to test the scalability of TreeNL and compare it with CARE and CURLER. Unless otherwise noted, we set the number of micro-clusters in CURLER to be 300 and set the maximum size of feature subset (same as our $fset_{max}$) in CARE to be 4 as default in this set of experiments. And the default setting of TreeNL (TreeNL$^+$) is,

- g_{min}: 5% of the data objects
- $fset_{max}$: 4

In Figure 4.16(a), we compare the runtime of CURLER, CARE, TreeNL and TreeNL$^+$ on a set of datasets containing 8000 $\sim$ 12000 objects. Each object has 40 features. Note that, for TreeNL and TreeNL$^+$, g_{min} is always 5% of the number of objects in the datasets. As we can see, their runtimes are comparable except for TreeNL. CURLER does not allow overlapping between clusters which makes it the fastest algorithm. CARE allows overlapping but only handles linear correlation clusters. Thus CARE is also slightly faster than TreeNL$^+$. The runtime of CURLER and CARE increases almost linearly with the number of objects. TreeNL$^+$'s runtime becomes slightly super-linear when the number of objects is large. The improvements enable TreeNL$^+$ to run in comparable speed to CARE and CURLER and much faster than the basic TreeNL.

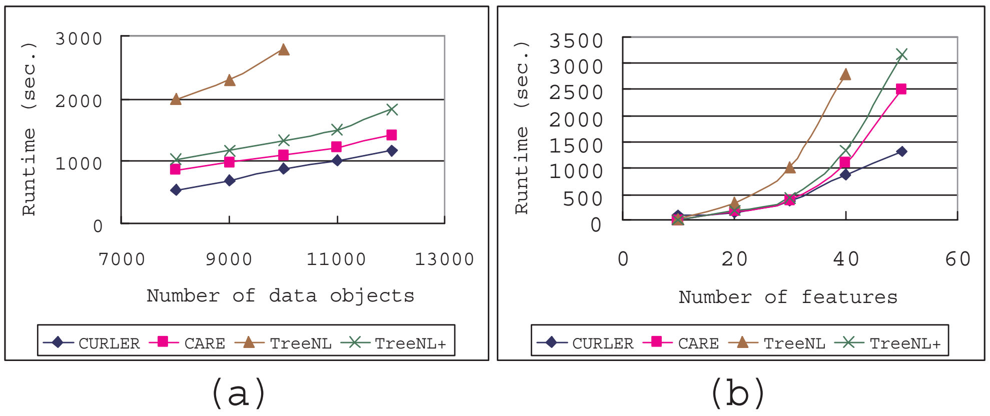


Figure 4.16: Runtime comparison of CURLER$_{300}$, CARE, TreeNL and TreeNL$^+$ on datasets of various sizes

In Figure 4.16(b), we compare the runtime of the four algorithms on a set of datasets containing 10 $\sim$ 50 features. All the datasets have 10000 objects. The runtimes are comparable among CURLER, CARE and TreeNL$^+$. CURLER's runtime still increases linearly with the number of features. CARE, TreeNL and TreeNL$^+$ have quadratic increase in runtime because they enumerate the feature subspaces. The improvements make TreeNL$^+$ much faster than TreeNL.

We also vary two of the input parameters of TreeNL, $fset_{max}$ and g_{min} on a 10000×30 synthetic data. In Figure 4.17(a), we plot the runtimes of TreeNL and TreeNL$^+$ when varying $fset_{max}$ from 2 to 5. Since CARE has the same input parameter, its runtime is also shown in Figure 4.17(a). All three algorithms have quadratic increase in runtime with the increase of $fset_{max}$. TreeNL$^+$ and CARE have similar runtime and are much faster than the basic TreeNL.

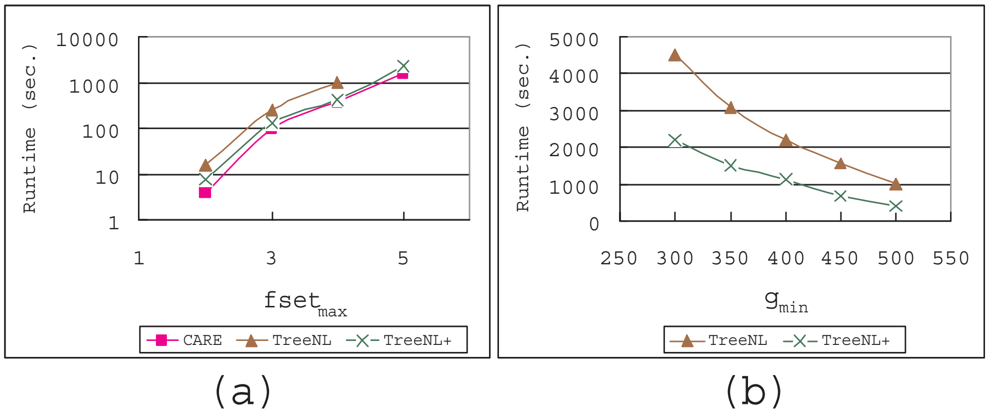


Figure 4.17: Runtimes of TreeNL and TreeNL$^+$ when varying $fset_{max}$ and g_{min}

We vary g_{min} from 300 (3%) to 500 (5%) on the same 10000×30 data. The runtimes are plotted in Figure 4.17(b). With smaller g_{min}, TreeNL and TreeNL$^+$ examine larger tree hierarchies and more implied partitions. Thus, the runtimes of TreeNL$^+$ and TreeNL have slightly super-linear increase when g_{min} decreases.

4.7 Conclusion

In this chapter, we extend the idea of TreeQA and propose a tree-based nonlinear correlation clustering algorithm, TreeNL. TreeNL enumerates the feature subspaces and utilizes tree hierarchies to calculate the correlation. The enumeration of feature subspaces provides the flexibility such that data objects are allowed to support multiple correlated feature subsets. And the tree hierarchy provides efficiency and effectiveness:

1. A tree hierarchy captures the distribution of the objects in the feature subspace and hence is able to reveal non-linear correlation clusters.

2. A tree hierarchy provides a multi-level resolution to examine the data objects.

We also define a novel correlation score using BIC (Bayesian information criterion) to avoid expensive operations such as computing matrix covariance and SVD. Our experiment results show that, compared with previous algorithms such as CURLER (Tung et al. (2005)) and CARE (Zhang et al. (2008)), TreeNL is more effective in detecting both linear and nonlinear correlation clusters with arbitrary overlapping. By efficient algorithm design and implementation, the runtime performance of TreeNL$^+$ is also comparable with that of CARE and CURLER.

Chapter 5

Sample Selection in Biallelic Data for Maximum Diversity

5.1 Introduction

In Chapters 2 - 4, I discussed the phylogeny-based genome-wide association mapping methods and their application in correlation clustering. From this chapter, I will focus on the second part of my thesis: maximum-diversity sample selection on both biallelic and non-biallelic data.

As mentioned in Chapter 1, the sample selection problem is closely related to the genome-wide association mapping problem. Genetic (allele) diversity is an important consideration when designing association mapping studies. The problem also accrues in other application domains such as customer review analysis and text mining.

A set's diversity cover can be viewed as a variation of the classical set cover problem where at least one example including and omitting each set element is required. Furthermore, it is useful to relax the requirement of a strict cover by specifying a minimal diversity threshold (usually specified as a percentage) that is to be retained by the selected subset. The implications and motivation for finding diversity subsets also varies between application domains.

Genetic Diversity

There are many experimental scenarios where the ultimate objective is to maintain, or at least maximize, genetic diversity within relatively small breeding populations. Examples include the design of breeding programs for livestock, the captive breeding of endangered species, and the construction of recombinant inbred lines for genetic mapping in animals and plants. Allele diversity is also an important consideration when designing association studies. In the case of genetic mapping in mice, there are several existing RIL panels (Williams et al. (2001)) with greater than 50 lines whose genotypes are known. Economics might dictate performing a pilot study across only a subset of the available lines (Xu et al. (2005); Jin et al. (2004)). The following question arises: What subset preserves that greatest diversity among a set of selected markers?

Low-cost genotyping technologies provide an important tool for measuring diversity at a biomolecular level in terms of Single Nucleotide Polymorphisms (SNPs). The knowledge of a SNP's presence, frequency, and location is leveraged in a wide range of experimental designs. By definition, a SNP must be present in a minimum frequency in a population (typically 5% in human studies). We consider a SNP to be lost if it is not represented within a population sample, and our goal is to minimize this loss.

It has been previously shown in (Ideraabdullah et al. (2004)) that over 99% of SNPs are biallelic, which enables us to represent alleles as a binary matrix. The approach to handle non-biallelic data will be discussed in Chapter 6.

Previously, pairwise phylogenetic distances were used to identify maximum genetic diversity subsets (Hartmann and Steel (2006); Steel (2005)). When applied to SNPs, this approach only considers the number of inconsistencies between column pairs in the allele diversity matrix, which is less information than the full matrix that our method considers.

Besides SNPs, gene expression values in other microarray data can also be used as a measurement for genetic diversity with proper discretization. And it is a similar problem to select a subset of the samples that preserves the greatest diversity among the genes.

Customer Diversity

In e-commerce, vendors often need to solicit customer opinions on the objects (e.g., products and/or services) they provide. This feedback is valuable for profiling customers, analyzing product preferences, and building recommendation systems. There is a practical limit on the number of objects that can be listed in a questionnaire. In fact, objects should be selected carefully to maximize information for subsequent analysis. That is, they should be a small number of informative (and unbiased) representatives of all available objects of interest. Intuitively, we want the selected objects to be non-redundant and cover the full range of customer's opinions (i.e., including both positive and negative ratings). The goodness of a selection can be measured by its customer-rating diversity coverage.

A small number of selected objects is also preferred for certain data modeling tasks, such as classification. We will show in the experiment section that subsets of objects, which cover large diversity, can be used to build better (more accurate and simpler) classifiers than the full object set.

The problem of finding an optimal diversity cover is NP-Complete. An interesting variant of this problem is to find an optimal diversity subset of a given size or smaller that achieves at least a given level of diversity. In this paper, we present practical algorithms for finding such optimal diversity subsets.

Our algorithm has two phases. In the first phase, a greedy approach is used to find an initial solution and establish an achievable bound in terms of subset size and coverage. Then in the second phase, an exhaustive search for all optimal subsets is systematically performed which is seeded with parameters derived from the initial greedy solution. We then employ pruning strategies to enable an efficient search for the globally optimal solution. Extensive experiments on real datasets from three applications demonstrate the effectiveness and efficiency of our algorithm.

5.2 Related Work

Many algorithms have been designed to establish a summary of a data matrix such that the maximum diversity is retained.

Co-clustering algorithms which cluster rows and columns of a data matrix simultaneously based on information theory are presented in (Chakrabarti et al. (2004); Dhillon et al. (2003b)). Dhillon (Dhillon et al. (2003b)) generates a flat partition of the data matrix into row and column clusters that maximizes the mutual information. The algorithm proposed by Chakrabarti (Chakrabarti et al. (2004)) partitions the rows and columns such that the sum of the entropy in each cluster is minimized. The sub-matrix generated by combining the rows and columns in each cluster found by these algorithms can be considered as a summary of the original data.

Feature selection has been used extensively in the classification literature (Dash and Liu (1997)). Given a class label for each row in the data matrix, the features that can maximize the classification accuracy are selected. Regardless of the diversity retention, the feature selection algorithms only select the most relevant features. These algorithms are based on heuristic methods and are not guaranteed to find an optimal solution.

The greedy solutions to the Set Cover problem and its variations are studied in (Hochbaum and Pathria (1998)). These algorithms find greedy solutions which maximize the coverage over the elements at each step. Greedy solutions are only guaranteed to be within a specific ratio of the optimal solution.

5.3 Preliminaries

In this section, we develop notations that will be used in this chapter and we provide formal problem definitions.

5.3.1 Diversity Cover (DC)

We use the same set of designations for data matrix, sample and marker as defined in Chapter 2. We assume that the data matrix H is binary. In a SNP data, "0" and "1" represent different alleles of a SNP marker m_i. While in the review data, $H(i,j) = 1$ represents that reviewer m_i ranks object s_j positively and $H(i,j) = 0$ represents that reviewer m_i ranks object s_j negatively.

An example of matrix H is shown in Table 5.1.

Table 5.1: A Sample-Marker Matrix

	s_1	s_2	s_3	s_4	s_5	s_6
m_1	0	1	1	1	0	0
m_2	1	1	0	0	0	1
m_3	0	1	1	1	0	0
m_4	1	0	1	0	1	0
m_5	1	0	0	0	1	1
m_6	0	0	0	1	0	0
m_7	0	0	0	0	1	0
m_8	0	0	0	0	0	1

Given a subset of samples, $S'_j \subseteq S$, the diversity of marker m_i is covered by S'_j if and only if there are two samples in S'_j, s_l and s_k , such that $H(i,l) = 1$ and $H(i,k) = 0$. For the sample subset S'_j, the total fraction of the covered markers is called the diversity coverage of S'_j, denoted as $C(S'_j)$, $0 \leq C(S'_j) \leq 1$. For example, given the matrix in Table 5.1, the sample subset $\{s_4, s_5\}$ has diversity coverage 0.75, $C(\{s_4, s_5\}) = 0.75$. Obviously, for any single sample, its coverage is 0. It is generally reasonable to assume that the coverage of the entire sample set is 1, $C(S) = 1$.

Now we define the Diversity Cover problem.

Diversity Cover (DC) Problem: Given a sample set S, a marker set M and a sample-marker matrix H, find the minimum subset D, $D \subseteq S$ such that $Coverage(D) = 1$.

For example, given the sample-marker matrix in Table 5.1, the minimum subset that covers all the markers is $\{s_4, s_5, s_6\}$.

5.3.2 Diversity Cover is NP-Complete

We show DC is NP-complete via a reduction from *set cover* (Cormen et al. (2001)). Given a collection of subsets, S, from the finite universal set, U, a set cover solution is the smallest group of subsets from S that covers all of U. Consider the following matrix construction for set cover. We associate each row with an element in U and each column with a subset in S. Each 1 in the matrix indicates an element's membership in a corresponding subset. Thus, set cover corresponds to finding the smallest subset of columns that provide a 1 in every row.

Suppose that we augment a set cover problem matrix with an additional row (element) and column (subset) with all 0 entries except for a single 1 at the intersection of the new row and column. This forces a DC solution to choose this newly added column. Moreover, if we ignore this added column, the remaining subsets are a solution to the original set cover problem. On the other hand, if given a solution to the original set cover problem, one can just add the last row to get a DC solution. Thus, DC is NP-complete.

5.3.3 Parameterized Diversity Cover (PDC)

In the Diversity Cover (DC) problem, we want to find the minimum subset that covers all markers. However, in some cases users are willing to lose the coverage of a few markers in order to find a smaller subset, e.g., in some SNP data, almost all the samples must be included to cover all the SNPs because of the large number of singleton SNPs (i.e., SNPs in which the rarer allele is present in a single strain). Therefore, we modify the DC problem by allowing a "minimum coverage ratio", ρ, rather than a full cover. Now we want to find the minimum subsets that covers no less than ρ.

Parameterized Diversity Cover (PDC) Problem: Given a sample set S, a marker set M and a sample-marker matrix H, find the minimum subset D (or subsets), $D \subseteq S$ such that $Coverage(D) \geq \rho$.

We can see that the DC problem is a special case of the PDC problem when the minimum coverage ρ is set to 1.

5.3.4 Upper Bound of Subset Coverage

Given a sample subset, $D, D \subseteq S$, the upper bound of its coverage can be calculated using the coverage of its subsets.

Property 5.3.1. *Given sample subset $D = \{s_1, s_2, ..., s_k\}, k \geq 3$, the upper bound of $C(D)$ is:*

$$C(D) \leq \frac{C(D - \{s_k\}) + C(D - \{s_{k-1}\}) + C(\{s_{k-1}, s_k\})}{2} \tag{5.1}$$

Proof: Let $D'=\{s_1, s_2, ..., s_{k-2}\}$, $D=D' \bigcup \{s_{k-1}\} \bigcup \{s_k\}$. Let X be the marker set covered by $D' \bigcup \{s_{k-1}\}$, Y be the marker set covered by $\{s_{k-1}\} \bigcup \{s_k\}$ and Z be the marker set covered by $D' \bigcup \{s_k\}$:

$$C(D' \bigcup \{s_{k-1}\}) = |X|/|M|, C(\{s_{k-1}\} \bigcup \{s_k\}) = |Y|/|M|,$$

$$C(D' \bigcup \{s_k\}) = |Z|/|M|.$$

Let W be the marker set covered by D . We have

$$W = X \bigcup Y \bigcup Z$$

For any marker m_l in Z, either it is already covered by D' or it is only covered when sample s_k is considered together with D'. In the first case, m_l also belongs to X since markers in X are covered by $D' \bigcup \{s_{k-1}\}$. In the second case, all the samples in D' have the same value on marker m_l and sample s_k has the opposite value on m_l. If sample s_{k-1} has the same value on m_l as s_k, m_l is also covered by $D' \bigcup \{s_{k-1}\}$ and belongs to X. Or if s_{k-1} has the opposite value on m_l compared to s_k, m_l is covered by $\{s_{k-1}\} \bigcup \{s_k\}$ and belongs to Y. Therefore we have

$$Z \subseteq X \bigcup Y \tag{5.2}$$

$$W = X \bigcup Y$$

$$C(D) = (|X| + |Y| - |X \bigcap Y|)/|M| \tag{5.3}$$

For any marker m_l in $X - Y$ which is covered by $D' \bigcup \{s_{k-1}\}$ but not by $\{s_{k-1}, s_k\}$, we know that m_l is either covered by D' alone or by D' together with $\{s_{k-1}\}$ and we know that s_k has the same value as s_{k-1} on m_l. Therefore, in either case, m_l is also covered by $D' \bigcup \{s_k\}$ and belongs to Z. Similarly, for any marker m_l in $Y - X$ which is covered by $\{s_{k-1}, s_k\}$ but not $D' \bigcup \{s_{k-1}\}$, we know that samples in D' have the same value as s_{k-1} on m_l while sample s_k has the opposite value to s_{k-1} on m_l. Therefore, m_l is also covered by $D' \bigcup \{s_k\}$ and belongs to Z. We can get

$$X - Y \subseteq Z, Y - X \subseteq Z \tag{5.4}$$

Since $(X - Y) \bigcap (Y - X) = \emptyset$, we have

$$|Z| \geq |X| - |X \bigcap Y| + |Y| - |X \bigcap Y| \tag{5.5}$$

According to Equations 5.3 and 5.5,

$$C(D) = \frac{|X| + |Y| - |X \bigcap Y|}{|M|} \leq \frac{|X| + |Y| + |Z|}{2|M|}$$

Therefore,

$$C(D) \leq \frac{C(D - \{s_k\}) + C(D - \{s_{k-1}\}) + C(\{s_{k-1}, s_k\})}{2}$$

□

When the subset D contains only 3 samples, the upper bound in Equation 5.1 becomes the exact value of $C(D)$.

Property 5.3.2. *Given the pair-wise diversity coverage of three samples, s_i, s_j and s_k,* *the coverage of set $\{s_i, s_j, s_k\}$ is known.*

$$C(\{s_i, s_j, s_k\}) = \frac{C(\{s_i, s_j\}) + C(\{s_i, s_k\}) + C(\{s_j, s_k\})}{2} \tag{5.6}$$

The proof is similar and is omitted for brevity. Obviously, by using Equations 5.1 and 5.6 recursively, we can establish an upper bound of subset coverage using only the pair-wise coverages.

Theorem 5.3.3. *Given a sample subset $D = \{s_1, s_2, ..., s_k\}$, we can calculate the upper bound of $C(D)$ using only the pair-wise coverage $C(\{s_i, s_j\})$, $s_i, s_j \in D$ according to Equations 5.1 and 5.6.*

For example, the upper bound of $C(\{s_1, s_2, s_3, s_4\})$ is

$$C(\{s_1, s_2, s_3, s_4\}) \leq [2C(\{s_1, s_2\})+2C(\{s_3, s_4\})+C(\{s_1, s_3\})+C(\{s_2, s_3\})+C(\{s_1, s_4\})+C(\{s_2, s_4\})]/4$$

Note that for a sample subset D, we can get several coverage upper bounds based on Theorem 5.3.3 by exchanging the order of the samples in D. We discuss the details of calculating a diversity upper bound using Theorem 5.3.3 in Section 5.4.2.

5.4 Algorithms

In this section, we present our Exhaustive Subset Enumeration (*ESE*) algorithm that solves the Parameterized Diversity Problem. Our algorithm guarantees to find all the minimum sample subsets that have diversity coverage no less than ρ. The *ESE* algorithm has two phases. In the first phase, a greedy algorithm, Parameterized Greedy Diversity Subset (*PGDS*), is used to find an initial sample subset S_G that has $C(S_G) \geq \rho$. Then in the second phase, we present an optimal K-ρ Diversity Subset ($K\rho DS$) algorithm to exhaustively search for all sample subsets with sizes K and smaller and with coverages no less than ρ. The initial sample subset S_G and several pruning strategies are used to reduce the searching space. The pseudocode of the *ESE* algorithm is shown in Figure 5.1.

Input:

- Sample Set S, Marker Set M, Sample-Marker Matrix H
- Minimum Diversity Coverage ρ

Output: A set of minimum sample subsets, I. $\forall S' \in I, C(S') \geq \rho$ **Method:**

```
1. $S_G$=$PGDS(\rho, S, M, H)$.
2. $K$=$|S_G|$.
3. $I$=$K\rho DS(K,\rho, S, M, H)$.
```

Figure 5.1: The *ESE* Algorithm

5.4.1 Parameterized Greedy Diversity Subset Algorithm

In Section 5.3, we proved that Diversity Cover is NP-complete and can be mapped to Set Cover. There is a well known greedy algorithm for the Set Cover problem. It chooses the subset that maximizes the increase in coverage in each step until all the elements are covered. The greedy algorithm can achieve an approximation ratio of $H(z)$, $H(z) = \sum_{k=1}^{z} \frac{1}{k} \leq \ln z + 1$ and $z = |M|$ Cormen et al. (2001).

In the first phase of *ESE*, we design a similar algorithm, Parameterized Greedy Diversity Subset (*PGDS*), to find greedy approximations to the Parameterized Diversity Cover problem. The *PGDS* algorithm also chooses the sample that maximizes the increase in the diversity coverage in each step. There are two differences in the *PGDS* algorithm compared with the greedy approach of the Set Cover problem.

1. The *PGDS* algorithm cannot pick the best first sample based on coverage because every single sample has zero coverage. Therefore, *PGDS* is restarted with each sample and we pick the smallest subset from the n generated subsets, where n is the number of samples.

2. The *PGDS* algorithm stops once the coverage of the sample subset exceeds the minimum threshold ρ.

The details of the *PGDS* algorithm are shown in Figure 5.2. The algorithm considers each sample in step 2, and the minimum subset among all the S' is reported as S_G. The time complexity of *PGDS* is $O(kn^2)$ where $k = |S_G|$.

Input:

- Sample Set S, Marker Set M, Sample-Marker Matrix H
- Minimum Coverage ρ

Output: sample subset S_G having $C(S') \geq \rho$

Method:

```
1. $S_G = \{\}$.
2. for all $s_i \in S, i = 1 \ldots n$.
3.     $S' = \{s_i\}$, $R = S - S'$, $c = 0$.
4.     while $c < \rho$
5.       $S' = S' \bigcup \{s_l\}$, which
                  $C(S' \bigcup \{s_l\}) = Max_{s_j \in R}(C(S' \bigcup \{s_j\}))$.
6.       $R = R - \{s_l\}$, $c = C(S')$.
7.     if $|S'| < |S_G|$
8.       $S_G = S'$.
```

Figure 5.2: The *PGDS* Algorithm

For example, if we are given the matrix in Table 5.1 and set $\rho = 1$, the subset found by *PGDS* is $S_G = \{s_1, s_4, s_5, s_6\}$ which is larger than the optimal minimum subset $\{s_4, s_5, s_6\}$.

5.4.2 Optimal K-ρ Diversity Subset Algorithm

In the first phase of the *ESE* algorithm, the *PGDS* algorithm finds an initial subset S_G satisfying $C(S_G) \geq \rho$. It establishes an upper bound on the size of the optimal subsets in I, i.e., any subset S' which has $C(S') \geq \rho$ should have size smaller than or equal to subset S_G. Let K=$|S_G|$, the exhaustive enumeration need to be performed only on the

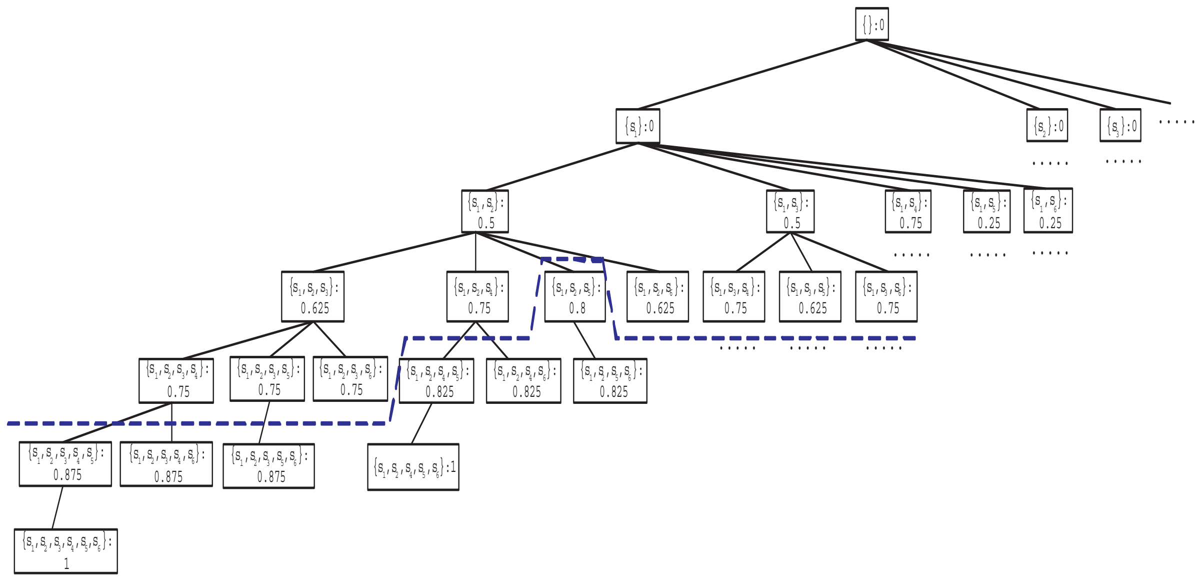


Figure 5.3: Enumeration Tree of the Matrix in Table 5.1

subsets having size no larger than K. The exhaustive enumeration can take exponential time in principle. However, with efficient pruning strategies, our enumeration algorithm, $K\rho DS$, performs much better in practice, finding the optimal subsets quickly.

The $K\rho DS$ algorithm searches all possible combinations of samples up to size K in an enumeration tree. Figure 5.3 illustrates part of the enumeration tree of the matrix in Table 5.1 and represents our search when we do not apply any pruning strategies. Each node in the tree stores a sample subset S' and the corresponding $C(S')$. The root represents the empty set. For each child node, the sample subset has one more sample than its parent node.

The $K\rho DS$ algorithm performs a depth-first search (Cormen et al. (2001)) on the enumeration tree. By imposing an order on the samples, the algorithm is able to perform a systematic search by enumerating all combinations, i.e., no combination is missed or revisited. Without loss of generality, let's assume the order is $s_1, s_2, \ldots, s_n$. For example, the depth-first search order on the enumeration tree in Figure 5.3 is $\{s_1, s_1s_2, s_1s_2s_3, s_1s_2s_3s_4, s_1s_2s_3s_4s_5, s_1s_2s_3s_4s_5s_6, s_1s_2s_3s_4s_6, s_1s_2s_3s_5, s_1s_2s_3s_5s_6, s_1s_2s_3s_6, \ldots\}$.

Among all the subsets that achieve the coverage threshold ρ, only the minimum sample subsets are reported. For example, if we set $\rho = 0.8$, only the nodes $\{s_1, s_2, s_3\}$ and $\{s_4, s_5, s_6\}$, which are below the dashed line in Figure 5.3, satisfy the ρ-threshold and subset-size constraints. Note that among all the nodes below the line, only those on the boundary need to be examined since nodes below the boundary have subsets of larger size. Details of the basic $K\rho DS$ algorithm are shown in Figure 5.4.

```
Input:
  • Sample Set $S$, Marker Set $M$, Sample-Marker Matrix $H$
  • Minimum Coverage $\rho$, Maximum Size $K$, $K = |S_G|$
Output: A set of minimum sample subset, $I$,
  $\forall S' \in I, C(S') \geq \rho, |S'| \leq K$
Method:
  1. Initialize.
       • Candidate minimum sample subset list, cList=$\emptyset$.
       • Current sample subset, cSample=$\emptyset$.
       • Remaining sample subset, rSample=$S$.
  2. Enumerate(cSample, rSample).
  3. $I$=the minimum subsets in cList.
Subroutine: Enumerate(cSample', rSample')
Method:
  1. if |cSample'| $\geq K$
  2.    return.
  3. for each $s_i \in$ rSample'
  4.    if $C$(cSample' $\bigcup \{s_i\}$)$\geq \rho$
  5.       Insert set cSample' $\bigcup \{s_i\}$ into cList.
  6.    else
  7.       cSample"=cSample' $\bigcup \{s_i\}$.
  8.       rSample"=rSample' - $\{s_i\}$.
  9.       Enumerate(cSample", rSample").
 10.    rSample'=rSample' - $\{s_i\}$.
```

Figure 5.4: The $K\rho DS$ Algorithm

The enumeration tree is dynamically materialized according to a depth-first searching order. At each node, the coverage of the corresponding sample subset S' is calculated based on the sub-matrix $H' = M' \times S'$, where M' is the set of markers that are not covered by the sample set in the parent node. The use of the dynamically generated sub-matrix can efficiently reduce the runtime of the $K\rho DS$ algorithm.

In the worst case, the $K\rho DS$ algorithm takes exponential time. In order to accelerate the search, we use several pruning strategies to reduce the search space.

Pruning Strategy 1: Dynamically Limit the Size of the Minimum Sample Subset

In the first phase of *ESE*, the greedy algorithm *PGDS* provides an upper bound of the size of the minimum sample subset. When the $K\rho DS$ algorithm searches the enumeration tree, it does not need to check any node of more than K samples.

The size of the minimum sample subsets can also be updated dynamically during enumeration. It is possible that K may be larger than the minimum size. The value of K is updated to be the size of the smallest subset S', satisfying $C(S') \geq \rho$, found so far. All remaining nodes representing larger subsets can be pruned from the enumeration tree without further examination. For example, if K is 9 and the algorithm finds a subset of 8 samples that can satisfy the threshold ρ, K is revised to 8 and any subsequent subsets of more than 8 samples are pruned from the enumeration tree.

Pruning strategy 1 is applied at step 5 of subroutine **Enumerate()** in Figure 5.4. When *cSample'* $\bigcup \{s_i\}$ is inserted into *cList*, its size is compared with that of the smallest subset in *cList*. If *cSample'* $\bigcup \{s_i\}$ is smaller, K can be updated accordingly and all the subsets in *cList* having larger size can be dumped.

Pruning Strategy 2: Order Samples by Pair-wise Coverage

For each node, we can estimate the increase in coverage for each sample from $rSample'$ based on its pair-wise coverage with every sample in $cSample'$. For example, in Figure 5.3, consider node $cSample' = \{s_1, s_2\}$, $rSample' = \{s_3, s_4, s_5, s_6\}$. We know that the pair-wise coverages are:

- $\mathbf{s_3}$: $C(\{s_1, s_3\}) = 0.5, C(\{s_2, s_3\}) = 0.25$
- $\mathbf{s_4}$: $C(\{s_1, s_4\}) = 0.75, C(\{s_2, s_4\}) = 0.25$
- $\mathbf{s_5}$: $C(\{s_1, s_5\}) = 0.25, C(\{s_2, s_5\}) = 0.75$
- $\mathbf{s_6}$: $C(\{s_1, s_6\}) = 0.25, C(\{s_2, s_6\}) = 0.5$

For each sample in $rSample'$, we use the sum of its pair-wise coverage with each sample in $cSample'$ as its score. This score is an (optimal) estimate of the additional coverage this sample can bring.

$$Score(s_3) = 0.75, Score(s_4) = 1, Score(s_5) = 1, Score(s_6) = 0.75$$

We sort, in descending order, the samples in $rSample'$ based on their scores so that in the sub-tree of node $cSample' = \{s_1, s_2\}$, sample s_4 and s_5 will be added first followed by s_3 and s_6. We can see that subsets having larger coverage are searched first in this case.

The sample sorting is conducted at each node dynamically. The pair-wise coverage of all samples can be calculated in advance and retrieved to compute the scores. At the root of the enumeration tree, the samples are initially sorted according to their order selected by the *PGDS* algorithm.

Pruning strategy 2 can be used before step 3 of subroutine **Enumerate()** in Figure 5.4. Samples in *rSample'* are sorted accordingly.

In some cases where the estimated size of minimum subsets, K, by *PGDS* is equal to or close to their actual size, pruning strategy 2 itself cannot reduce the search space dramatically. However, when combined with the following pruning strategy, it always delivers a substantial improvement in efficiency.

Pruning Strategy 3: Estimate a Branch Upper Bound on Coverage

The coverage of sample subsets generally increases monotonically when adding new samples. For each node in the enumeration tree, we can calculate an upper bound, $C(cSample' \bigcup rSample')$, on the coverage of any sample subsets represented in the subtree. Any subsets represented in the branch must have coverage no larger than that value. We call it the *branch-upper-bound*. For example, consider node $\{s_1, s_3, s_5\}$ in Figure 5.3, $cSample' = \{s_1, s_3, s_5\}$ and $rSample' = \{s_6\}$. The upper bound is $C(\{s_1, s_3, s_5, s_6\})$, which is 0.825. If the *branch-upper-bound* of the subtree is less than the minimum coverage threshold ρ, we can safely prune the subtree.

It is inefficient to calculate the upper bound at each node independently by adding up the samples in $cSample'$ and $rSample'$. However, we can calculate it simultaneously with the depth-first search by tracking the samples that are absent in the subtree under each node.

Given a node with its $cSample'$ and $rSample' = \{s_{i_1}, s_{i_2}, ..., s_{i_q}\}$, its left-most child node has the same *branch-upper-bound*. Let $cSample''_1$ and $rSample''_1$ be the current and remaining samples at the left-most child node. We have

$$cSample''_1 \bigcup rSample''_1 = (cSample' \bigcup \{s_{i_1}\}) \bigcup (rSample' - \{s_{i_1}\})$$

$$= cSample' \bigcup rSample'$$

where s_{i_1} is the first sample in $rSample'$.

For the j^{th} child nodes $(1 < j \leq q)$ of the current node, we have

$$cSample''_j \bigcup rSample''_j = (cSample''_{j-1} \bigcup rSample''_{j-1}) - \{s_{i_{j-1}}\}$$

Therefore, we can calculate the *branch-upper-bound* of a node according to the upper bound of its parent node or its siblings. The *branch-upper-bound* at the root node is 1 since every sample appears in some nodes. When we proceed along a branch, this value decreases as more samples are absent in the sub-tree. For example, if we know that the upper bound at node $\{s_1, s_3\}$ in Figure 5.3 is 1,

- for its child node $\{s_1, s_3, s_4\}$, the upper bound is still 1 because $\{s_1, s_3, s_4\}$ is the left-most child node of $\{s_1, s_3\}$.
- for $\{s_1, s_3, s_5\}$, sample s_4 is absent, its upper bound becomes 0.875.
- for $\{s_1, s_3, s_6\}$, sample s_4 and s_5 are absent. Its branch upper bound on coverage becomes 0.75.

Pruning strategy 3 can be used before step 4 of subroutine **Enumerate()** in Figure 5.4. If the upper bound on branch coverage is less than ρ, the subroutine can stop and return to its previous level.

As mentioned earlier, pruning strategy 2 can improve the efficiency of pruning strategy 3. After we sort the succeeding samples at each node in the tree, the last several branches are likely to be pruned by strategy 3 because they contain only those samples that have the least increase in coverage. Our experiments on real datasets suggest that using pruning strategies 1 and 3 together reduces the runtime of the $K\rho DS$ algorithm by $70\% - 80\%$. Combining pruning strategies 1, 2 and 3 can reduce the runtime by more than 95%.

Pruning Strategy 4: Refine the Branch Upper Bound on Coverage

In pruning strategy 3, we estimate the *branch-upper-bound* using the current sample subset and all its succeeding samples in *rSample'*. This upper bound is loose because in many cases, we cannot include all the succeeding samples into the current subset. For example, if the current node represents a subset of p samples and there are q succeeding samples in *rSample'*, we can at most include a subset of $K - p$ samples from *rSample'* during the search in the subtree under the current node. If we can calculate the maximum increase in coverage after adding any subset of $K - p$ samples from *rSample'*, we get a tighter upper bound than the one in pruning strategy 3.

Suppose that the current sample subset is $cSample = \{s_{i_1}, s_{i_2}, ..., s_{i_p}\}$ and the succeeding samples are $rSample = \{s_{j_1}, s_{j_2}, ..., s_{j_q}\}$. $cSample$ covers the marker subset M_a, $M_a \subset M$, and the uncovered marker subset is M_b, $M_b = M - M_a$. Since marker set M_b is uncovered by $cSample$, all the samples in $cSample$ have the same value on each marker in M_b. Therefore, we can use a dummy sample s_{j_0} to represent the diversity of $cSample$ on M_b. When adding a subset of samples from $rSample$, S', into the current subset $cSample$, the increase of coverage is the coverage of $S' \bigcup \{s_{j_0}\}$ on M_b. We can calculate the pair-wise coverage on M_b between any two samples in $\{s_{j_0}, s_{j_1}, s_{j_2}, ..., s_{j_q}\}$. Let the set of pair-wise coverage be $C_{pair} = \{c_1, c_2, ..., c_m | m = (q + 1)q/2\}$. Note that coverage is still calculated based on $|M|$, so that the total coverage on M can be calculated by adding the coverage on M_b (increase of coverage) and the coverage on M_a (current coverage) together.

In order to know the maximum increase in coverage after adding any subset of $K - p$ samples from *rSample'*, we need to calculate the upper bound of the coverage of any $(K - p)$ samples from $rSample$ together with s_{j_0} on M_b. However, in order to make the problem easier, we loosen the requirement and calculate the upper bound of the coverage of any $K - p + 1$ samples of *rSample'*$\bigcup\{s_{j_0}\}$ on M_b.

According to Theorem 5.3.3 in Section 5.3, by recursively applying Equation 5.1, we can get the upper bound of the coverage of any u samples using their pair-wise coverage. The upper bound should be in the following form

$$C_{max} \leq \sum_{i=1}^{(u-1)u/2} a_i \cdot C(s_j, s_k), a_1 \geq a_2 \geq a_3... \tag{5.7}$$

Now we have $q+1$ samples in total and we know the set of all their pair-wise coverage C_{pair}, we can calculate the upper bound of the coverage of any subset of $K-p+1$ samples by replacing $C(s_j, s_k)$ in Equation 5.7 with the $\frac{(K-p)(K-p+1)}{2}$ largest pair-wise coverage in C_{pair}. If the pair-wise coverage in C_{pair} are sorted in descending order, the upper bound of increasing coverage after adding $K-p$ samples into *cSample* is

$$\triangle C \leq \sum_{i=1}^{(K-p)(K-p+1)/2} a_i \cdot c_i, c_i \in C_{pair} \tag{5.8}$$

Let C be the current coverage, $C = \frac{|M_a|}{|M|}$. If $C + \triangle C$ is still less than ρ, the $K\rho DS$ algorithm does not need to search the subtree under the current node because there is no sample subset in the subtree that is not larger than K in size and with coverage not less than ρ.

Equation 5.8 provides a tighter upper bound than the one in pruning strategy 3 especially when there are large number of samples in the data. However, in order to get the upper bound, pair-wise coverage on M_b between $\{s_{j_0}, s_{j_1}, s_{j_2}, ..., s_{j_q}\}$ must be computed. Note that the coefficients a_i in Equations 5.7 and 5.8 are constants and can be calculated for each size of sample sets in advance. The computation and the pruning can be inserted before step 7 of subroutine **Enumerate()** in Figure 5.4. We can see that pruning strategies 3 and 4 are used in different places in the algorithm. In fact, these two strategies can be used together though strategy 4 provides tighter upper bound. Pruning strategy 3 is much faster than strategy 4 and, therefore, it is used as the pre-pruning step before pruning with strategy 4.

Though calculating the pair-wise coverage of $\{s_{j_0}, s_{j_1}, ..., s_{j_q}\}$ at each node takes time, we demonstrate in our experiments that the extra time used for coverage calculation is negligible compared with the runtime saved by pruning branches using pruning strategy 4. Also, as a side product, the actual increase in coverage for adding each sample from *rSample'* into *cSample'* is known during the pair-wise coverage calculation. Therefore, in pruning strategy 2, instead of ordering the samples by the estimated score, the algorithm can now order the samples in *rSample'* by their actual increase in coverage.

5.5 Experiments

In this section, we present results on synthetic and real data to show the efficiency of our algorithms and the effectiveness of the selected sample subsets. One real dataset is a SNP panel from recombinant inbred mouse strains. The other two real datasets are of customer review type.

Data

- Perlegen data[1]: The Perlegen dataset contains genotypes from 15 commonly used laboratory mouse strains[2], {129S1/SvImJ, A/J, AKR/J, BALB/cByJ, BTBR T+ tf/J, C3H/HeJ, CAST/EiJ, DBA/2J, FVB/NJ, KK/HlJ, MOLF/EiJ, NOD/LtJ, NZW/LacJ, PWD/PhJ, WSB/EiJ} and a reference strain (C57BL/6J). These 16 strains account for over 85% of all inbred strains used in biomedical research. The dataset contains $8,322,543$ SNPs in total. The dataset is imputed using the method described in (Roberts et al. (2007)).

- Congressional Voting Records data[3]: The voting dataset includes votes from 435 Congressmen on 16 key votes. The votes can be 'yes' or 'no' and are denoted by 1 and 0. The 435 congressmen are classified into two groups, 267 democrats and 168 republicans.

- Jester data[4]Goldberg et al. (2001): The Jester dataset contains 4.1 Million ratings (-10.00 to +10.00) of 100 jokes from 73,421 users. We discretize the data by replacing positive ratings by 1 and negative ratings by 0. Since none of the 73,421 users completes the review for all the 100 jokes, we use jokes that were reviewed

[1]http://mouse.perlegen.com/mouse/index.html

[2]We regard each mouse strain as a sample.

[3]http://www.ics.uci.edu/ mlearn/MLSummary.html

[4]http://www.ieor.berkeley.edu/ goldberg/jester-data/

by more than 70% of the users and then select users who reviewed all these jokes. Thus, the dataset we use contains 46,268 users and 30 jokes without missing values.

- Synthetic data: The synthetic data is randomly generated. The dataset is a binary matrix consisting of $40,000$ rows and 100 columns. We consider the rows as markers and columns as samples.

The synthetic dataset is mainly used to demonstrate the efficiency of our algorithms. And the three real datasets are mainly used to demonstrate the effectiveness of the selected sample subsets.

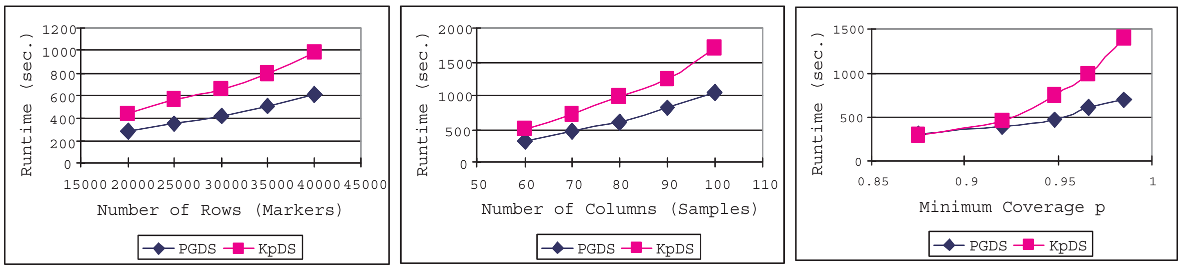
Except as otherwise noted, we use all the four pruning strategies collectively in the experiments because this combination provides the best runtime performance. The algorithms are implemented in MATLAB, and all experiments are conducted on a PC with CPU P4 3GHz, 1G RAM and 80G HDD.

5.5.1 Efficiency Analysis

In this section, we demonstrate the efficiency of the *PGDS* and the $K\rho DS$ algorithm using the synthetic data and some of the real datasets.

Scalability

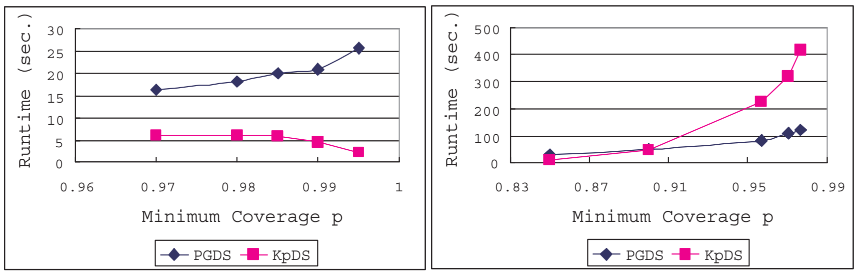
For the synthetic data, we vary the number of rows, the number of columns and the minimum coverage ρ respectively. The default values for these settings are: number of rows=40k, number of columns=80 and minimum coverage ρ=0.965. While we are varying one of the settings, the other two use the default values. The runtime performance of *PGDS* and $K\rho DS$ is shown in Figure 5.5. The runtime of both algorithms increases linearly when the number of rows increases in Figure 5.5(a). And the runtime increases quadratically when the number of columns and ρ increase for both algorithms as shown in Figure 5.5(b) and (c).



(a) Varing number of Rows (b)Varing number of columns (c)Varing minimum coverage

Figure 5.5: Scalability: Runtime on Synthetic Data

For the real datasets, we only vary the minimum coverage ρ setting and use all the rows and columns. Both *PGDS* and $K\rho DS$ can finish searching the Voting data in 1 second. Therefore, we only show the runtime performance on Perlegen and Jester data in Figure 5.6.



(a) Perlegen Data (b) Jester Data

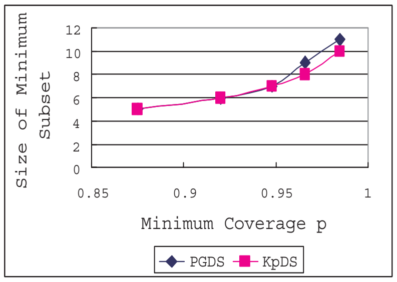
Figure 5.6: Scalability: Runtime on Real Data

The runtime performance on the Jester data is similar to that of the synthetic data for both algorithms. For Perlegen data, $K\rho DS$ can even be faster than *PGDS* because of the pruning strategies. Also, the runtime of $K\rho DS$ begins to drop when ρ is larger than 0.985. The reason is that the entire searching space becomes smaller when $\rho > 0.985$. The minimum subsets have size larger than 9 when $\rho > 0.985$ and causes the shrinking of the entire searching space because a subset of 10 samples or more contains more than half of the samples in the data.

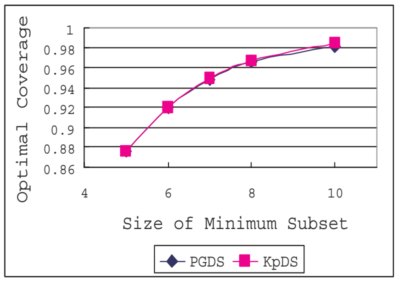
Note that the total runtime of the *ESE* algorithm is the sum of the runtime of *PGDS* and $K\rho DS$.

Comparison of Subsets Found by *PGDS* and $K\rho DS$

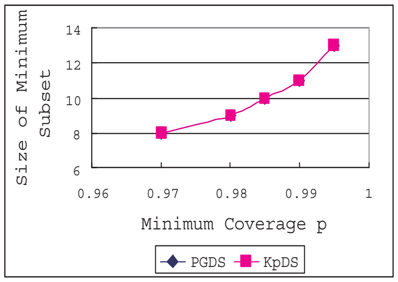
As we discussed in Section 5.4, the sample subsets found by *PGDS* may not be the optimal subset, i.e., either there exists a smaller subset that can achieve the minimum coverage ρ or there exists a subset with the same size but has larger coverage. Thus, in this section, we compare the subsets found by *PGDS* and $K\rho DS$. In the first part of the experiments, we vary minimum coverage ρ and compare the size of the minimum subsets found by both algorithms. Then we use the set of sizes of the minimum subsets found by $K\rho DS$ in the first part, and compare the optimal coverage that is achieved by the two algorithms for each of the subset size. The results are shown in Figure 5.7.



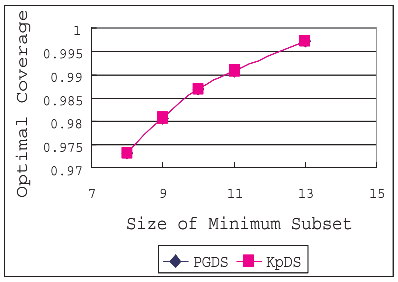
(a) Synthetic Data



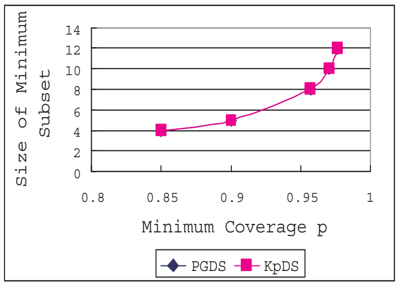
(b) Synthetic Data



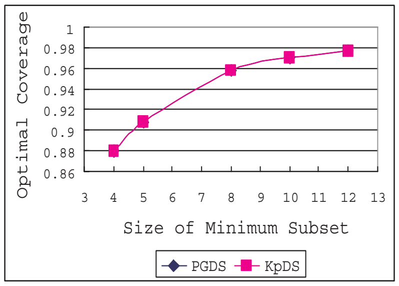
(c) Perlegen Data



(d) Perlegen Data



(e) Jester Data



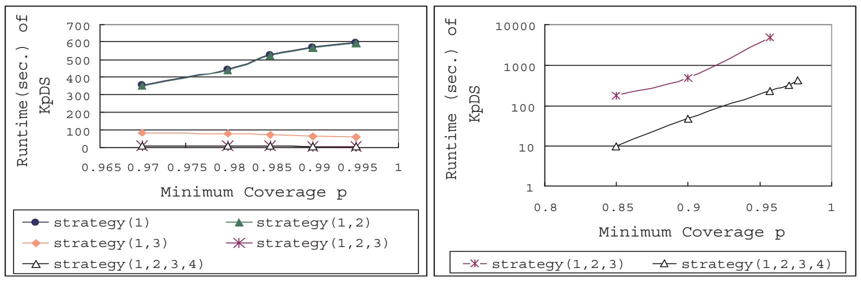
(f) Jester Data

Figure 5.7: Comparison of Subsets

As we can see, on the synthetic dataset, *PGDS* finds a larger subset when ρ becomes large. And for subsets of size 8 and 10, the true optimal subset found by $K\rho DS$ has larger coverage than the subset found by $PGDS$. However, as shown in Figure 5.7(c-f), $PGDS$ always find the same optimal subset as $K\rho DS$ on the real datasets. The result on the Voting data is the same as the Perlegen and Jester data and is omitted.

Efficiency of Pruning Strategies

In this section, we compare the efficiency of the pruning strategies discussed in Section 5.4. We vary the minimum coverage parameter ρ and compare the runtime performance on Perlegen and Jester data. The synthetic data is not used because it is too large for $K\rho DS$ to search without any one of the pruning strategies. And the Voting data is too small to be used to show the difference.



(a) Perlegen Data (b) Jester Data

Figure 5.8: Efficiency of Pruning Strategies

Figure 5.8(a) shows the results on the Perlegen data. We observe that pruning strategies (1,2,3) and pruning strategies (1,2,3,4) give the best performance, which is orders of magnitude faster than other strategy combinations. The reason that pruning strategy 4 does not improve the performance significantly when used in combination with strategies 1,2 and 3 is that the Perlegen dataset only contains 16 samples. This is sufficiently small that the two upper bounds from strategies 3 and 4 are close to each other. Using only pruning strategies 1 and 2 (sorting) just slightly reduces the runtime. This is because sorting only helps the $K\rho DS$ algorithm find minimum subsets faster, but cannot reduce the search space by pruning sub-trees. Using only pruning strategies 1 and 3 saves about $70\% - 80\%$ of the runtime of enumerating with strategy 1 only. Without sorting, strain subsets offering good coverage are randomly distributed in the enumeration tree. Strain sorting helps to bring these branches together in the enumeration tree so that effective pruning can be achieved.

On the Jester data, the $K\rho DS$ algorithm can finish the tasks in reasonable time only with pruning strategies (1,2,3) or pruning strategies (1,2,3,4). And for pruning strategies (1,2,3), it also can not afford a minimum coverage ρ larger than 0.96. As we can see, pruning strategies (1,2,3,4) are orders of magnitude faster than pruning strategies (1,2,3). As we discussed in Section 5.4, pruning strategy 4 has a large advantage over pruning strategy 3 when the number of samples becomes large.

5.5.2 Effectiveness Analysis

In this section, we apply our algorithm on the three real datasets and demonstrate the effectiveness by analyzing the selected sample subsets.

Perlegen Data

The Perlegen dataset has 16 samples and more than $8M$ SNPs. As we discussed in Section 5.1, in the design of recombinant inbred lines, an important measurement for a set of lines (samples) is its diversity coverage on the SNPs. A subset of 8 strains was hand selected for the Collaborative Cross (Churchill et al. (2004)) by biologists based on the phylogenetic relationships assumed for strains. We compare this subset with the best 8-strain subsets found by the *ESE* algorithm in Table 5.2.

As we can see, the *ESE* subset achieves higher coverage than the Collaborative Cross subset. Four strains are common to both subsets: **129S1/SvImJ**, **CAST/EiJ**, **PWD/PhJ** and **WSB/EiJ**. Aside from **129S1/SvImJ**, all strains are wild-derived from the three major Mus musculus subspecies. We plot the distribution of the diversity coverage of random set of 8 samples in Figure 5.9. The coverage of the Collaborative

Table 5.2: Perlegen Data: Comparing the 8-strain subsets of the Collaborative Cross with the maximum diversity solution found by *ESE*

		Coverage
Collaborative Cross Subset	**129S1/SvImJ**, **CAST/EiJ**, **PWD/PhJ**, **WSB/EiJ**, NZW/LacJ, C57BL/6J, NOD/LtJ, A/J	0.8926
ESE Subset	**129S1/SvImJ**, **CAST/EiJ**, **PWD/PhJ**, **WSB/EiJ**, KK/HlJ, DBA/2J, MOLF/EiJ, FVB/NJ	0.9575

Cross subset, 0.8926, is labelled by the red dotted line in the figure. The Collaborative Cross subset has coverage larger than more than 70% of the randomly selected 8-sample subsets while the *ESE* subset is obviously the one has the largest coverage.

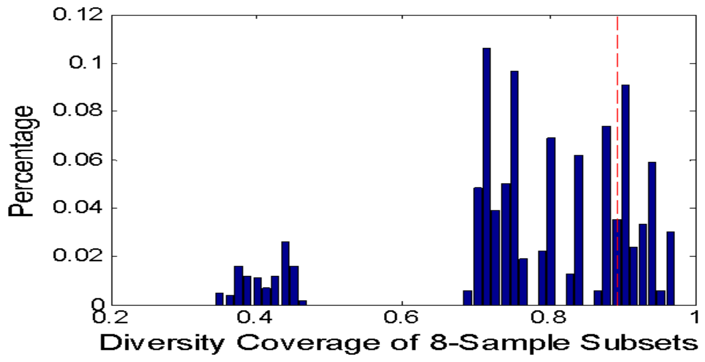


Figure 5.9: Perlegen Data: Distribution of Diversity Coverage of 8-sample Subsets

Voting Data

The Voting data includes votes of the 435 Congressmen on 16 key votes. We consider the congressmen as markers and the key votes as samples. Our *ESE* algorithm finds two sample subsets that consist of 5 samples and have diversity coverage = 1, i.e., all the markers (congressmen) are covered. The two subsets are listed in Table 5.3.

Table 5.3: Voting Data: Subsets of 5 Samples found by *ESE* that have coverage = 1

Subset 1	handicapped-infants, physician-fee-freeze religious-groups-in-school, mx-missile, duty-free-exports
Subset 2	physician-fee-freeze, el-salvador-aid anti-satellite-test-ban, mx-missile, synfuels-corporation-cutback

As we discussed in Section 5.1, these subsets can be used to generate simpler yet more accurate classification models. We use Weka [5], which is a data mining software in Java, to build different classifiers based on all the 16 samples and the two 5-sample subsets. Classification accuracy is calculated by using 10-fold cross-validation. The accuracy of the classifiers are listed in Table 5.4.

Table 5.4: Voting Data: Accuracy of Classifiers based on full set and subsets

Classifiers	Full Sample Set	Subset 1	Subset 2	Random Subset
RandomTree	92.8%	**95.4**%	95.17%	86.66%
PART	95.4%	95.17%	**95.86**%	86.89%
NaiveBayes	90.11%	**94.25**%	93.33%	86.66%
KStar	92.87%	**94.25**%	93.56%	86.66%
BFTree	95.4%	95.17%	**95.86**%	87.12%
NBTree	95.4%	95.4%	**95.86**%	86.66%
SMO(SVM)	**95.86**%	95.63%	95.63%	87.12%

As shown in Table 5.4, except for SMO(SVM), the highest accuracy always occurs in one of the subsets found by *ESE* for all the other classifiers. As expected, the randomly selected subset which also consists of 5 samples always has the lowest accuracy. Moreover, the decision trees built by NBTree on Subsets 1 and 2 are much simpler than that of the full sample set because of the smaller number of samples. The trees are omitted here for space restriction.

Jester Data

We discussed in Section 5.1 that subsets of samples can also be helpful in designing customer review study. By applying our *ESE* algorithm on the Jester data, we get many sample (joke) subsets that are small and cover most markers (reviewers). Given the minimum coverage ρ, the number of qualified sample subsets and their sizes are listed in Table 5.5. The sample (jokes) subsets in Table 5.5 suggest that reviewers' ratings on a small number of objects are sufficient to retain most diversity.

[5]http://www.cs.waikato.ac.nz/ml/weka/

Table 5.5: Jester Data: Number of qualified sample subsets and their sizes for given ρ

ρ	size	number of qualified subsets
0.9	5	122
0.95	8	73
0.97	10	34

According to the experiment results we presented in this section, we demonstrated that our algorithms are both efficient and effective.

5.6 Conclusion

In this chapter, we introduced the Parameterized Diversity Cover problem: given a sample-marker dataset and a minimum coverage threshold ρ, find the minimum sample subset that achieves coverage ρ. We propose an efficient exhaustive subset enumeration algorithm (*ESE*) which can find the optimal solution. The algorithm has two stages: (1) a greedy approach, *PGDS*, is used to first find an approximate solution for minimum subset with coverage no less than ρ; (2) an enumeration algorithm, $K\rho DS$, then searches for the optimal solution in the enumeration tree using several pruning strategies. We have evaluated the performance on three real datasets.

Chapter 6

Representative Sample Selection in Non-biallelic Data for Maximum Diversity

6.1 Introduction

In Chapter 5, we presented the *PDGS* and $K\rho DS$ algorithms that can find the minimum subsets which retain at least $\rho\%$ of the diversity in biallelic data. The coverage property of the biallelic data reduces the searching space of the $K\rho DS$ algorithm and speeds up the algorithm significantly.

Given a non-biallelic data, we face the following challenges:

- We can not define the diversity in non-biallelic data by simple coverage (as defined in Chapter 5).
- The exhaustive searching scheme no longer works because the coverage property (Property 5.3.1, Chapter 5) does not apply in non-biallelic data.

In non-biallelic data, we designate the selected maximum-diversity samples as the **Representative Set**. A good representative set should capture the most information from the original dataset compared to other subsets of the same size. Also, it should have low redundancy. Algorithms such as Maximum Coverage (S.Hochbaum and Pathria

(1998)) can generate a subset that captures original information from a dataset, but may only work well in a balanced dataset, where the number of transactions from each class is similar. However, the maximum coverage approach does not generate good representative sets that take into consideration low redundancy. Good performance of the maximum coverage approach depends on an appropriate choice of similarity function and similarity threshold.

General cluster algorithms address the problem to some extent, especially representative-based clustering algorithms such as the k-medoid clustering (Kannan et al. (2000)). However, as we will show in the experiment section, generating a representative set in advance can help the processing of representative-based clustering algorithms.

In this chapter, we model the diversity in non-biallelic data using information-theoretic measures, mutual information and relative entropy (Cover and Thomas (1991); Hastie et al. (2001)). To meet the expectation that the representative set should capture the most information and avoid redundancy, we design an objective function and a greedy algorithm, *REP*, to make the optimal choice at each step when selecting a new representative. We also design a simplified version of the greedy algorithm which employs heuristics to achieve much better performance.

6.2 Related Work

LIMBO (Andritsos et al. (2003)) is an hierarchical clustering algorithm based on Information Bottleneck framework. It produces a compact summary model of the data in the first and then employs Agglomerative Information Bottleneck(AIB) algorithm to work on the summarized data. By summarizing the data, LIMBO can handle larger dataset than AIB can.

In (Slonim and Tishby (2000)), a two-phase clustering algorithm is designed for document clustering. The algorithm first performs clustering on words, and then on documents, using the generated word clusters. Its runtime complexity is around $O(mn^2)$, where m is the number of required clusters and n is the size of dataset. While our method takes only $O(mn)$.

Storyline (Kumar et al. (2004)) is an approach for clustering web pages using graphic theorem. It builds a bipartite document-term graph and figures out each dense sub-bipartite graph which is actually a set of closely related pages and terms and can be summarized into a cluster. One problem with this method is that though it can cluster web pages into groups, it may not find a proper representative for each group.

Max Coverage (S.Hochbaum and Pathria (1998)) can handle the problem we studied in this chapter by selecting samples which are similar to most of the samples in the dataset. However Max Coverage cannot capture original information as much as *REP* since it only considers coverage while omitting redundancy.

In (Basu et al. (2004)), a semi-supervised clustering method based on information theory performs clustering using predefined constraints. However, to get better performance, the algorithm tends to require more constraints which may be difficult to generate manually.

In (Dhillon et al. (2003a)), a word clustering algorithm replaces the classical feature selection method on document-words datasets. In (Dhillon et al. (2003a)), words are clustered in a supervised way. Instead of using mutual information between words and documents, it maintains mutual information between words and classes.

6.3 Preliminary

We present some information-theoretic measurements in this section. Since we have two requirements for a good representative: high coverage and low redundancy, we employ two information-theoretic measurements. We use mutual information to measure the coverage of the representatives; good representatives that capture most information in the original dataset should have a large mutual information value with respect to the features of the dataset. We will use relative entropy to measure the redundancy between the representatives. A high relative entropy between representatives infers a low redundancy. Therefore, as we will see in the next section, our objective function will consist of two parts which are equally important. We will define terms and provide examples related to mutual relative entropy and mutual information in this section.

A non-biallelic data matrix is represented by $H = S \times F$, in which S represents the sample set and F represents the feature set[1].

	f_1	f_2	f_3	f_4	f_5
e_1	1	1	1	1	0
e_2	0	2	2	2	0
e_3	1	0	0	1	1
e_4	0	0	3	3	3
e_5	1	0	0	0	1

(a) Example Data

	f_1	f_2	f_3	f_4	f_5
e_1	0.25	0.25	0.25	0.25	0
e_2	0	0.33	0.33	0.33	0
e_3	0.33	0	0	0.33	0.33
e_4	0	0	0.33	0.33	0.33
e_5	0.5	0	0	0	0.5

(b) Distribution Matrix

Figure 6.1: Example Data

Given an example data in Figure 6.1(a), by normalizing each row in the matrix, we can view a sample as a distribution in the feature domain. Figure 6.1(b) shows the distribution matrix after normalizing the data in Figure 6.1(a).

We define two random variables, B and A, in the sample domain and feature domain respectively. Giving equal weight to each sample $s_i \in S$, we define:

$$p(B = s_i) = \frac{1}{|S|}, s_i \in S$$

According to the distribution table, we obtain the conditional probability $P(A|B)$. For example, $P(A = f_1|B = s_1) = 0.25$. For convenience, we use $P(f_1|s_1)$ to represent $P(A = f_1|B = s_1)$. For each subset S_i of S, we define the probability distribution

[1]M is used in Chapter 5 to represent the biallelic marker (feature) set. While in this chapter, H represents more general non-biallelic data. Therefore, we use F to represent the feature set. (F is used to represent the feature set in Chapter 4 for the same reason.)

function in the following way:

$$P(S_i) = \frac{|S_i|}{|S|}, S_i \subseteq S$$

$$P(A|S_i) = \sum_{s \in S_i} \frac{P(s)}{P(S_i)} P(A|s) = \frac{1}{|S_i|} \sum_{s \in S_i} P(A|s)$$

For two sample subsets, the relative entropy can be used to measure the difference. It is defined on the two corresponding probability distributions.

Definition 6.3.1. Relative entropy between sample subsets: Given two element subsets S_i and $S_j \subseteq S$, the relative entropy between them is the relative entropy or *Kullback-Leibler* divergence between their distributions in the feature domain:

$D_{KL}[S_i||S_j] = D_{KL}[P(A|S_i)||P(A|S_j)] = \sum_{f \in F} P(f|S_i) log \frac{P(f|S_i)}{P(f|S_j)}$

To avoid the problem of indefinite value when $P(f|S_i)$ or $P(f|S_j)$ equals 0, we will use a real number close to 0 to replace 0 in implementation such as 10^{-10}. In the discussion below, we will also use the relative entropy between two samples where single sample is considered as a degeneracy of sample subset.

A representative is a typical sample of the set S. *REP* aims to find a small representative set R from a huge collection of samples. We give a general definition of the representative set as follows:

Definition 6.3.2. Samples related to Representative: Given a representative r, a sample s is **related** to r if $D_{KL}(r||s) < (min_{s_i \in S-\{r\}} D_{KL}(r||s_i)) * t_{max}$, where $t_{max} \geq 1$, that is, the relative entropy between r and s is within a certain range of the minimal relative entropy between r and all other samples in S. t_{max} is a parameter used to control the range. We use $L(r)$ to denote the set of samples related to r. When $t_{max} < 1$, $L(r) = \{r\}$.

In principle, a sample may be related to several representatives which will make the problem complicated and make trouble on the random variable W which we will define later. Therefore, we make some modification on Definition 6.3.2 to resolve this issue. We generate representatives one by one, so that when we pick samples related to a new representative, we only consider those samples that are not related to any previously chosen representatives. By doing so, each sample will be related to at most one representative. Similar approach was used in some max coverage approaches.

Definition 6.3.3. Representative Set

A representative set R is a subset of S. For each representative $r_i \in R$, we can get its related element set $L(r_i)$, $L(r_i) \subseteq S$. Given a representative set $R = \{r_1, r_2, ..r_n\}$, $S = L(r_1) \cup L(r_2) \cup .. \cup L(r_n) \cup S_\theta$. S_θ contains all the samples which are not related to any representative in R.

In Definition 6.3.3, S_θ contains all the samples not related to any representative. For convenience in explaining *REP*, we consider S_θ the related set of a special representative r_θ that does not exist in the dataset, $L(r_\theta) = S_\theta$.

We define a random variable W over the representative set and r_θ. Given a representative set $R = \{r_1, r_2, ..r_n\}$,

$$P(W = r_i) = \frac{|L(r_i)|}{|S|}, P(W = r_\theta) = \frac{|S_\theta|}{|S|}$$

$$P(A|W = r_i) = \frac{1}{|L(r_i)|} \sum_{s \in L(r_i)} P(A|s)$$

$$P(A|W = r_\theta) = \frac{1}{|S_\theta|} \sum_{s \in S_\theta} P(A|s)$$

For convenience, we will use $P(f_1|r_1)$ to represent $P(A = f_1|W = r_1)$ later.

Mutual information is a measure of the relationship between two random variables. We can use mutual information between random variables W and A, $I(W, A) = H(A) - H(A|W)$, to measure the information captured when representing the original dataset with the representative set R. Intuitively, $I(W, A)$ measures how much variation in the feature domain A is captured by a representative set. The higher, the better. Given two representative sets R_1 and R_2, $R_1 = \{r_1, r_2, .., r_n\}$, $R_2 = R_1 \cup \{r_{n+1}\}$, and their corresponding $S_{1\theta} = L(r_{1\theta})$ and $S_{2\theta} = L(r_{2\theta})$, we get $L(r_{2\theta}) = L(r_{1\theta}) - L(r_{n+1})$. Using this equality, we can calculate the difference between $I(W_1, A)$ and $I(W_2, A)$:

$$\Delta I(W_2, W_1) = I(W_2, A) - I(W_1, A) = H(A|W_1) - H(A|W_2)$$
$$= \frac{|L(r_{2\theta})|}{|S|} D_{KL}[p(A|r_{2\theta})||p(A|r_{1\theta})] + \frac{|L(r_{n+1})|}{|S|} D_{KL}[p(A|r_{n+1})||p(A|r_{1\theta})]$$

Since relative entropy is always positive, we know that R_2 retains more information than R_1.

Property 6.3.1. (Monotonicity) *Given a representative set R, if we generate a new representative set R' by adding a new representative to R, we can always have $I(W', A) \geq I(W, A)$. W is the random variable defined over R and $\{r_\theta\}$. W' is the random variable defined over R' and $\{r'_\theta\}$.*

Table 6.1: Notations

S	the entire sample set, $S = \{s_1, s_2 ... s_n\}$
s, s_i	single sample, $s, s_i \in S$
S_i	subset of S, $S_i \subseteq S$
F	the entire feature set, $F = \{f_1, f_2 ... f_m\}$
f, f_i	single feature, $f, f_i \in F$
R	the representative set, $R \subseteq S$
r, r_i	single representative, $r, r_i \in R$
$L(r_i)$	set of elements related to representative r_i $L(r_i) \subseteq S$
S_Θ	set of elements not related to any representative in R, $S_\Theta \subseteq S$
r_Θ	the virtual representative for S_Θ, $L(r_\Theta) = S_\Theta$
B	random variable over domain of S
A	random variable over domain of F
W	random variable over domain of $R \cup \{r_\Theta\}$

6.3.1 Objective Function & Problem Definition

Property 6.3.1 suggests that we may use a greedy algorithm to successively pick representatives that offer the highest mutual information. Starting from an empty set $R = \emptyset$, each time we add a new representative to R which can increase the mutual information most – meaning it can capture more original information than any of the remaining non-representative elements. At the same time, we should also minimize the redundancy between the new representative and existing representatives. We measure the redundancy between two representatives by their relative entropy. High relative entropy infers big difference between the probability distribution of the two representatives and thereby small redundancy. Combining these two factors, we define our objective function as follows:

$$f(r_{new}, R) = \Delta I(W_{new}, W) + min_{r \in R}(D_{KL}(r_{new}||r))$$

The formal definition of our problem is as follows.

Problem Definition: Given a dataset which consists of samples $S = \{s_1, s_2, ..., s_n\}$, and an empty representative set R, add k representatives into R one by one such that at each step, the objective function $f(r_i, R)$ can be maximized.

6.4 Algorithms

In this section, we will first describe the greedy algorithm, *REP*, which generates the representative set. And then, we will give a simplified version of *REP*.

6.4.1 The *REP* Algorithm

REP utilizes a greedy scheme to select new representatives at each step which can maximize the objective function f until it gets the required number of representatives.

A formal description of *REP* is given in Algorithm 6.1. As we can see, the greedy algorithm is simple and easy to implement.

Algorithm 6.1 *Greedy Algorithm: REP*

```
Input: Dataset $H = S \times F$, Size of representative set, $k$.
Output: Representative Set $R$
1: $R = \{\}$
2: while $|R| < k$ do
3:     for all $s_i \in S_\theta$ do
4:         calculate $f(s_i, R)$
5:     end for
6:     $R = R \cup \{s\}$ if $f(s, R) = max_{s_i \in S_\theta}(f(s_i, R))$
7:     update $S_\theta$
8: end while
```

6.4.2 Simplified *REP*

The *REP* algorithm in the previous section has a computational complexity of $O(k|S|)$, where k is the number of representatives, and $|S|$ is the size of the sample set. When the dataset grows, it becomes time-consuming to generate the representative set. In applications in which response time is crucial, such as web search, we need to generate the representative set much faster.

As we look through Algorithm 6.1, we can find that the cause for the complexity is that at each iteration, we consider each remaining sample in the set as a candidate for the next representative. So if we can narrow the candidate set, we can expect a faster performance. According to our objective function in Section 6.3.1, a good representative maximizes information gain and dissimilarity with other representatives. While it may be difficult to estimate information gain in advance, it is easy to find samples dissimilar to the representatives already found. Since we have calculated the relative entropy between each pair of samples as part of preprocessing, we can use those results to find a set of samples which are most dissimilar to each representative very quickly. We can build the candidate set by taking the union of these dissimilar sets. Similar to Definition 6.3.2, we can define a dissimilar set for each representative.

Algorithm 6.2 *Simplified REP Algorithm*

```
Input: Dataset $H = S \times F$, Size of representative set, $k$.
Output: Representative Set $R$
1: $R = \{\}$
2: $Candidate = \{\}$
3: while $|R| < k$ do
4:    for each $r_j \in R$ do
5:       $Candidate = Candidate \bigcup D(r_i)$
6:    end for
7:    $Candidate = Candidate - \bigcup_{r_i \in R} L(r_i)$
8:    for all $s_i \in Candidate$ do
9:       calculate $f(s_i, R)$
10:   end for
11:   $R = R \cup \{s\}$ if $f(s, R) = max_{s_i \in Candidate}(f(s_i, R))$
12: end while
```

Definition 6.4.1. Dissimilar set of a representative

A sample s belongs to the dissimilar set of a representative r if and only if $D_{KL}(r||s) > (max_{s_i \in E-\{r\}} D_{KL}(r||s_i)) * t_{min}, t_{min} < 1$. We denote the dissimilar set of representative r as $D(r)$. Parameter t_{min} is used to control the size of the dissimilar set. A smaller t_{min} will result in a larger dissimilar set for a representative.

Definition 6.4.2. Candidate set for next representative

Given a set of generated representatives $R = \{r_1, r_2, \ldots, , r_k\}$, the candidate set for the next representative is: $Candidate = \cup_{i=1..k} D(r_i)$

In fact, the candidate set can be defined in a more general way. With a pre-defined integer x, the candidate set consists of samples which are contained in at least x dissimilar set of representatives. If $x = 1$, candidate set is the union of the dissimilar sets as we defined in Definition 6.4.2. And if x is the number of representatives, the candidate set is the intersection of the dissimilar sets. In our algorithm, we will use $x = 1$.

The simplified *REP* algorithm is described in detail in Algorithm 6.2. We can expect the algorithm to be faster and less optimal when the parameter t_{min} increases. As we will see in the experiment section, $t_{min} = 0.9$ is a proper value.

6.5 Applications and Experiments

In this section, we verify the effectiveness of the representative set. We apply the *REP* algorithm to different kinds of real-life datasets including the Mushroom dataset and the 20 Newsgroup dataset. All the experiments are conducted on a PC with PIV 1.6G CPU, 512M main memory and 30G hard drive. The algorithms are implemented in C.

Algorithms

We compare the performance of our algorithm against two others, **MaxCover** and **RandomPick**.

MaxCover is a greedy approach for Maximum k-coverage in (S.Hochbaum and Pathria (1998)). This approach assumes that every sample in the dataset has a coverage set which consists of samples similar to it. In our implementation, we define the coverage set of a sample in the same way as Definition 6.3.2.

Definition 6.5.1. Coverage set of a sample

The coverage set of sample s, C(s), is defined as $C(s) = \{s_i | D_{KL}(s||s_i) < (min_{s_i \in E-\{s\}} D_{KL}(s||s_i)) * c_{max}\}$, $c_{max} \geq 1$. c_{max} is a similarity threshold that is analogous to t_{max}.

In the RandomPick method, we randomly pick a subset of the dataset as representatives. The average performance of 10 runs is reported for each experiment.

Measurements

We use two measurements in experiments: *coverage* and *accuracy*. Coverage measures the percentage of classes that are covered by the representative set. A class is covered if and only if at least one of the representative belongs to that class. Let C(R) be the distinct number of class labels covered by representative set R and $|C|$ be the total number of classes in the dataset, then *coverage* is defined as:

$$coverage = \frac{C(R)}{|C|}$$

Besides the *coverage* measurement, we want to design a more rigid task to show the effectiveness of our representative set. Therefore we design a clustering algorithm using the representative set. Given a representative set, we obtain the class label of each representative[2]. Each remaining sample is assigned to the class of its closest representative. The description of the algorithm follows:

Algorithm 6.3 *Clustering Based on Representative Set*

```
Input: Dataset $H = S \times F$
Output: Clustering of the dataset
1: Generate representative set R, $|R| = k$, $k << |S|$
2: retrieve label for each representative in R
3: for all $s \in S$ do
4:     calculate $D_{KL}(r_i||s)$, for $\forall r_i \in R$
5:     assign $s$ to representative $r$ if
       $D_{KL}(r||s) = min_{r_i \in R}(D_{KL}(r_i||s))$
6: end for
```

We argue here that a good set of representatives would have the same class label as those samples that are being covered by them. Let $C(S)$ be the number of samples that have the same class label as their nearest representative. Then *clustering accuracy* is given in the form of:

$$\textit{clustering accuracy} = \frac{C(S)}{|S|}$$

For convenience, we will denote this measurement as *accuracy* in later discussions.

6.5.1 Mushroom Dataset

We use the Mushroom dataset from UCI machine learning archive. It contains 8124 elements and 22 categorical attributes. The elements are in two classes.

We vary the number of representatives from 2 to 10 and compare the coverage. We set the similarity threshold t_{max} and c_{max} to 4. The result is in Figure 6.2(a). Both *REP* and MaxCover cover the two classes when enough representatives are generated. However, *REP* does it faster than MaxCover and RandomPick.

[2]Both datasets in our experiments have class labels.

$\|R\|$	*REP*	MaxCover	RandomPick
2	100%	50%	70%
3	100%	100%	95%
4	100%	100%	90%
5	100%	100%	100%
10	100%	100%	100%

(a) Representative Coverage

$\|R\|$	*REP*	MaxCover	RandomPick
2	67.9%	51.7%	48.3%
4	75.1%	71.0%	63.5%
8	89.0%	89.2%	79.3%
20	96.3%	96.4%	90.7%
30	100%	96.3%	93.7%

(b) Clustering Accuracy

Figure 6.2: Mushroom dataset, $t_{max} = 4$, $c_{max} = 4$

We also compare the clustering accuracy achieved by the three methods in Figure 6.2(b). As we can see, the *REP* algorithm gives the best performance. MaxCover is better than RandomPick, however, since it does not consider the redundancy of the samples selected, it still performs worse than the representative set method.

Though MaxCover and *REP* are comparable in terms of *coverage* and *accuracy*, the reliable performance of MaxCover depends on a well-defined similarity threshold while the representative set method is much less sensitive to it. Small adjustment of c_{max} may result in poor performance, as shown in Figure 6.3(a) and 6.3(b). MaxCover fails to pick any samples from the second class until the 10^{th} representative and gets poor *accuracy*.

$\|R\|$	*REP*	MaxCover
2	100%	50%
5	100%	50%
10	100%	100%

(a) Representative Coverage

$\|R\|$	*REP*	MaxCover
8	89.0%	51.8%
20	98.5%	86.5%
30	100%	89.3%

(b) Clustering Accuracy

Figure 6.3: Mushroom dataset, $t_{max} = 3$,$c_{max} = 3$

Comparisons with other clustering algorithms

Several other algorithms have been applied on the Mushroom dataset. One of them is the SUMMARY algorithm (Wang and Karypis (2004)). This method summarizes the dataset by clustering it into several groups. When SUMMARY has 30 clusters generated, it achieves accuracy of 99.6%. And it does not get 100% accuracy until more than 400 clusters are generated. As we can see in Table 6.2, our *REP* algorithm can capture the information of the original dataset more efficiently and quicker than SUMMARY can.

Table 6.2: Clustering Accuracy on Mushroom Dataset, Compared with SUMMARY, $t_{max=4}$

REP		SUMMARY	
$\|R\|$	accuracy	$\sharp$ clusters	accuracy
30	100%	30	99.6%
50	100%	140	99.93%
...	...	298	99.99%
...	...	438	100%

In comparison with unsupervised clustering methods such as LIMBO (Andritsos et al. (2003)), *REP* also performs better. In (Andritsos et al. (2003)), the reported accuracy on the Mushroom dataset is about 91%. While in *REP*, specialists only need to check around 30 elements among 8000 elements to achieve the perfect result. The cost of manual processing is small relative to the improvement in accuracy.

Comparison of *REP* with its simplified version

In this section, we compare the performance of *REP* with that of its simplified version on the Mushroom dataset. Note that *REP* is a special case when parameter t_{min} is set to 0 in the simplified version. Therefore, we denote *REP* as a simplified version with $t_{min} = 0$ in this section.

First, we compare their runtime on the dataset. The preprocessing takes about 290 seconds and we exclude that from the figure below since the results are repeatedly used in different runs. As we can see in Figure 6.4, the simplified *REP* offers bigger performance improvement as t_{min} increases. The two curves of $t_{min} = 0$ and $t_{min} = 0.85$ are close to each other and exhibit similar trend while the curve of $t_{min} = 0.9$ is far below them. The curve of $t_{min} = 0.9$ even converges to a constant value after 40 representatives are identified when $t_{max} = 4$. This can be explained by looking into the number of candidates generated in each iteration. In Figure 6.5, we plot the number of candidates in each iteration. The curves of $t_{min} = 0$ and $t_{min} = 0.85$ are always close to each other while that of $t_{min} = 0.9$ is lower. On curve $t_{min} = 0.9$ when $t_{max} = 4$, the number of candidates drops dramatically in the last several iterations, which brings down the slope of the runtime growth and makes it logarithmic in Figure 6.4.

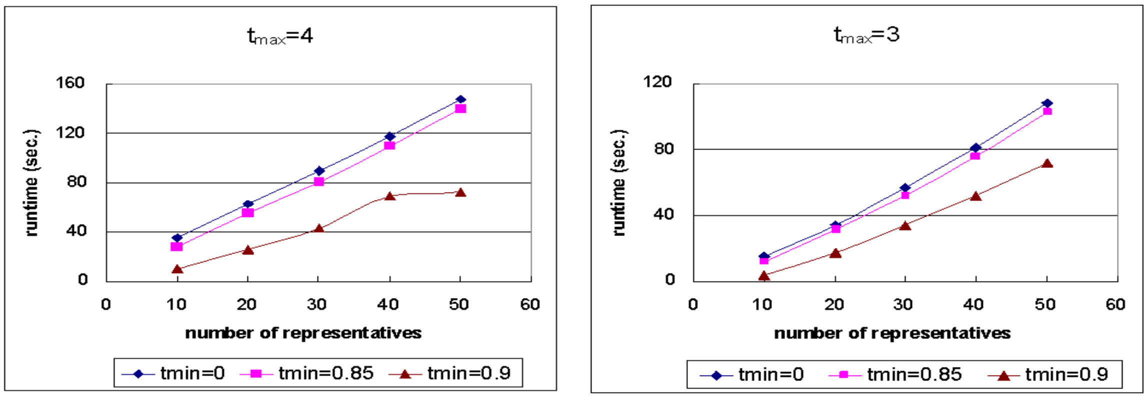


Figure 6.4: Runtime of different t_{min}

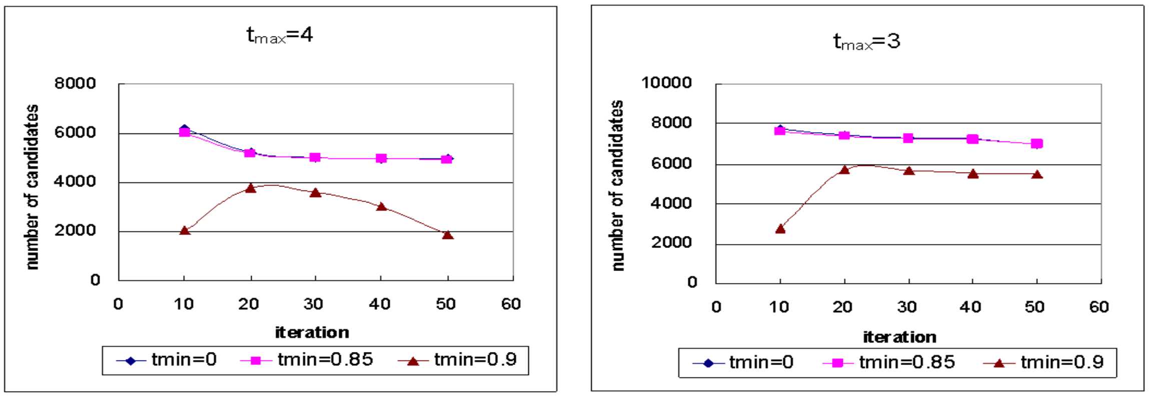


Figure 6.5: Number of candidates of different t_{min}

Besides runtime, we also compare the accuracy of the clustering algorithm based on the representative sets generated under different t_{min} values. As we can see from Figure 6.6, when $t_{min} = 0.85$, the performance is the same as $t_{min} = 0$ while at $t_{min} = 0.9$, the performance degrades slightly but is still much better than MaxCov, SUMMARY and LIMBO. This result confirms our discussion in Section 6.2.

$\lvert R \rvert$	$t_{min} = 0$	$t_{min} = 0.85$	$t_{min} = 0.9$
10	89.2%	89.2%	79.6%
20	96.3%	96.3%	96.3%
30	100%	100%	98.9%
40	100%	100%	99.9%
50	100%	100%	99.9%

(a) Clustering Accuracy, $t_{max} = 4$

$\lvert R \rvert$	$t_{min} = 0$	$t_{min} = 0.85$	$t_{min} = 0.9$
10	89.2%	89.1%	79.1%
20	98.5%	98.5%	98.3%
30	100%	100%	98.8%
40	100%	100%	99.9%
50	100%	100%	99.9%

(b) Clustering Accuracy, $t_{max} = 3$

Figure 6.6: Different t_{min} on Mushroom Dataset

6.5.2 20 Newsgroup Dataset

The 20 Newsgroup dataset is a document-words dataset. It consists of 20,000 newsgroup articles from 20 different newsgroups. Since there are more than 30,000 distinct words in all the articles, we conduct a scoring processing which is mentioned in (Slonim and Tishby (2000)). The top 2000 words with the highest score are selected as features.

We use three subsets of the entire 20 Newsgroup dataset to test our algorithm. Two of the subsets contain articles from two and three newsgroups respectively. Since we get similar results as the Mushroom dataset on these two subsets, we won't present the detailed results of them in this chapter. Interested readers can refer to technical report (Pan et al. (2005b)).

The third subset is the mini 20 newsgroup dataset which is a reduced version of the full 20 newsgroup dataset. It consists of the same set of 20 newsgroup topics, but each topic contains only 100 articles. We want to test the performance of the three algorithms with respect to the complexity of the data. In this case, the number of newsgroups included in the dataset is a good indicator of the data complexity. Because of the different characteristics of the samples in this mini 20 newsgroup dataset, we will set $t_{max} < 1$ and $c_{max} = 1.1$ in all the following experiments in this section.

First, we compare the methods on the mini 20 newsgroup data. The results are in Figure 6.7(a) and 6.7(b). As we can see, in both accuracy and coverage, *REP* outperforms the other two methods.

	Coverage		
$\|R\|$	*REP*	MaxCover	RandomPick
20	70%	55%	65%
40	85%	80%	88.5%
60	100%	90%	92%
80	100%	95%	100%
100	100%	100%	99%

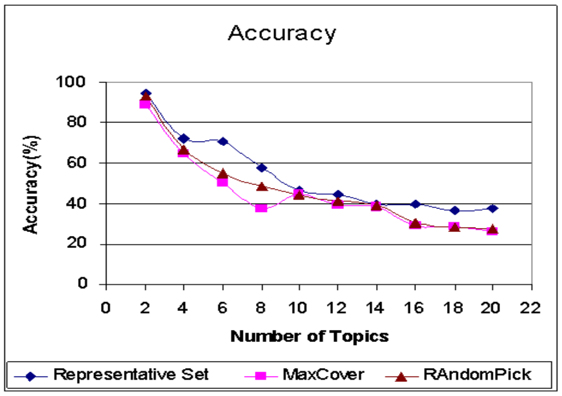
(a) Representative Coverage

	Accuracy		
$\|R\|$	*REP*	MaxCover	RandomPick
20	23.8%	12.5%	18.3%
40	32.5%	21.2%	21.7%
60	37.5%	26.1%	27.2%
80	38.8%	30.3%	28.8%
100	41.6%	32.6%	29.0%

(b) Clustering Accuracy

Figure 6.7: Mini 20 newsgroup, $t_{max} < 1$, $c_{max} = 1.1$

In order to show the change of performance by dataset containing different number of topics(classes), we start with a subset of the mini 20 Newsgroup consisting of 2 topics and add two topics into the dataset each time until it includes all 20 topics. For each of these dataset, we generate 60 representatives to study the accuracy and coverage. The changes of performance are shown in Figure 6.8.



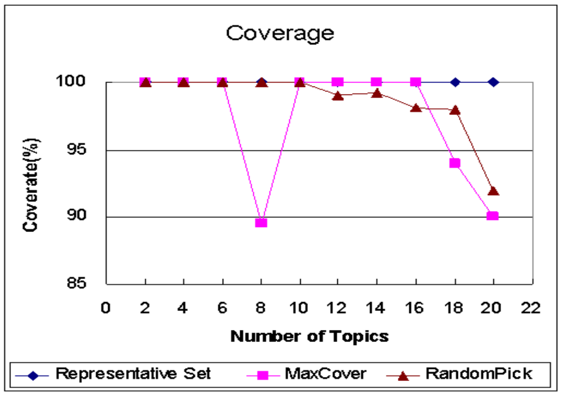


Figure 6.8: Performance of 60 representatives

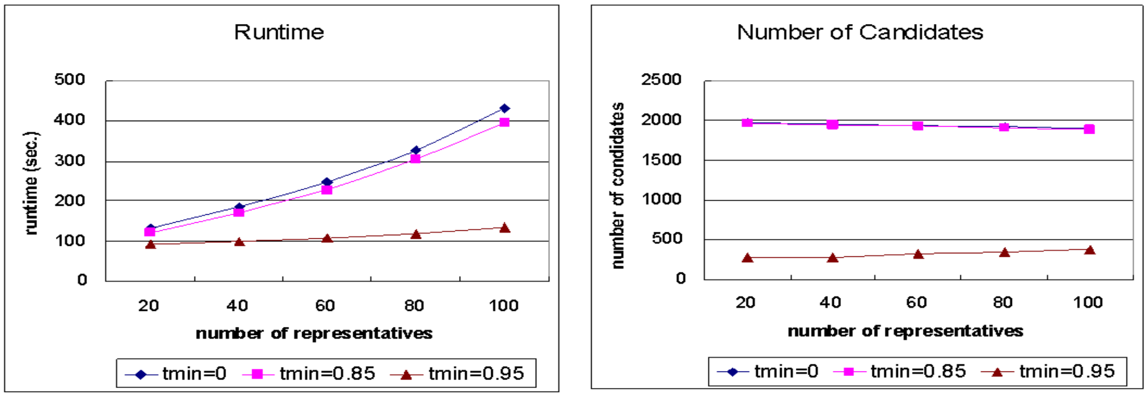
All three methods exhibit degrade *accuracy* when more topics are added into the dataset. However *REP* is always better than the other two. The *accuracy* of MaxCover and RandomPick get close when number of topics is large because each sample is similar to a set of other samples and the size of the similar set has a small deviation when there are large number of topics in the dataset. The choice made by MaxCover is then close to random.

For *coverage*, our algorithm maintains the same performance while other two methods fail to cover all topics when the number of topics increases. The MaxCover method has a big drop on coverage when 8 topics are included. This is because of the characteristic of the 8-topic subsets, i.e., several similar topics are included. And while 2 more topics are added in, the characteristic of the new subset changes.

Comparison of ***REP*** and its simplified version

As in the previous section, we will compare the runtime performance of our algorithms by varying parameter t_{min}.

We set t_{min} to 0, 0.85 and 0.95 to show its effects. As we can see in Figure 6.9(a), when we set t_{min} to 0.95, runtime drops dramatically. That is because when $t_{min} = 0.95$, the size of the candidate set for each iteration is small, which can been seen in Figure 6.9(b).



(a) Runtime (b) Number of Candidates

Figure 6.9: Performance of different t_{min}

Besides runtime, we also compare the goodness of the representative sets generated under different t_{min} by the clustering algorithm. We present the accuracy in Table 6.3.

Table 6.3: Accuracy of different t_{min}

$\lvert R\rvert$	$t_{min} = 0$	$t_{min} = 0.85$	$t_{min} = 0.95$
20	23.8%	23.8%	23.1%
40	32.5%	32.5%	31.5%
60	37.5%	37.5%	33.3%
80	38.8%	38.8%	34.5%
100	41.6%	41.6%	38.2%

When t_{min} is set to 0.85, the simplified *REP* achieves the same accuracy as the original *REP*. And when t_{min} is set to 0.95, its accuracy is slightly worse than the original algorithm, however, the results are still better than that of the MaxCov and RandomPick methods as in Figure 6.7(b). The slight degrade in accuracy brings the significant improvement in runtime as shown in Figure 6.9(a).

In all the experiments above, our representative set method always outperforms MaxCover and RandomPick. This shows the effectiveness of our representative sets.

6.6 Conclusion

In this chapter, we have defined a maximum-diversity subset, the representative set, in non-biallelic data. A representative set is a small subset of the original dataset, captures most original information compared to other subsets of the same size and has a low redundancy. We first design a greedy algorithm, *REP*, to generate the representative set. Then we build a simplified version based on the greedy algorithm for faster and better performance. Our experiments show that the representative set attains the desired characteristics and captures information more efficiently than other methods.

Chapter 7

Discussion

With the development of high-throughput and low-cost sequencing technologies, there has been an urgent need to develop highly efficient and scalable analysis methods for various genetic studies. In my thesis, I developed algorithms for the following two genetic analysis tasks on large genomic data:

- Genome-wide Association Mapping Study (GWA).
- Maximum-diversity Sample Selection.

Genome-wide association mapping aims to find the correlation between a target phenotype and the genetic markers. While many statistical models/tests have been developed for examining the correlation, the problem is still non-trivial due to its high-computational cost. Because of the complexity of the biological and environmental factors which regulate the phenotype, many existing methods (Zollner and Pritchard (2005); Mailund et al. (2006); Sevon et al. (2006)) use complex models which have higher statistical power but lower runtime efficiency. In my thesis, I developed two phylogeny-based methods, TreeQA and TreeQA+, for genome-wide association mapping. Both methods infer the phylogenies of the samples and incorporate the phylogenies into the association analysis. As demonstrated in the experiments, TreeQA and TreeQA+ are more effective in association mapping than the previous single-marker and haplotype-based methods (Pe'er et al. (2006); Akey et al. (2001); McClurg et al. (2006); Onkamo et al. (2002); Wang and Paigen (2005)) because of incorporating phylogenies into the model. On the other hand, I developed several efficiency optimizations in TreeQA and TreeQA+ so that they can handle genome-wide analysis efficiently. The scalability experiments show that TreeQA, TreeQA+ and the previous single-marker and haplotype-based methods have comparable runtime performance. I also developed the TreeNL algorithm which extends the idea of TreeQA and is applied to correlation clustering problems.

The second part of my thesis focuses on the maximum-diversity sample selection problem. Sample selection is closely related to genome-wide association mapping. Sample selection can be used to design the breeding program which produces the data for GWA. It can also be used to pre-process the genomic data and make GWA more efficient. In my thesis, I developed the *PGDS* and $K\rho DS$ algorithms to do sample selection in biallelic SNP data. The problem is NP-complete. The *PGDS* algorithm takes a greedy searching scheme which is fast but does not guarantee an optimal solution. And the $K\rho DS$ algorithm searches for the optimal solution exhaustively. It utilizes several pruning techniques to speedup the process. Genomic data can also be non-biallelic. I developed the *REP* algorithm to select samples in non-biallelic data. The *REP* algorithm also uses greedy searching but reports the near optimal solutions in most cases.

Experimental results in my thesis show that these algorithms tackle the GWA and sample selection problems efficiently and effectively. In the following sections, I discuss how these algorithms can be improved in the future work.

7.1 Genome-Wide Association Mapping

In Chapters 2 and 3, the phylogeny-based association mapping methods (TreeQA and TreeQA+) have only been applied to data produced by isogenic mouse strains. In fact, association mapping can also be conducted on other types of data, e.g., congenic strains and outbred strains. These datasets have very different characteristics from the isogenic strain data. Thus, they pose two problems for applying phylogeny-based genome-wide association mapping:

1. How do we infer phylogenetic trees from the various datasets? If we can only infer imperfect or near perfect phylogenetic trees (Sridhar et al. (2006); Satya et al. (2006)) from the data, how can we examine the association using these trees?
2. Can we combine the analysis results from the various datasets? Or can we run analysis simultaneously on the heterogeneous datasets?

I address the two problems in the following two sections respectively.

7.1.1 Phylogeny-based Association Mapping in the Various Biological Data

TreeQA and TreeQA+ infer perfect phylogenies from phased SNP data which are biallelic, such that each marker only has two alleles among the samples. However, other biological data can be ternary or just non-biallelic so that the perfect phylogeny inference method can not be applied. For example,

- In the unphased SNP data, each marker has up to three alleles. In this case, the compatible intervals (as defined in Chapter 2) may not exist because of the unphased genotypes. In (Ding et al. (2008)), an entropy-minimizing method is used to infer the phase of the data at first and then local phylogenies are built from the inferred haplotypes.

- In the microsatellite genotype data, each marker can have more alleles than the phased/unphased SNP data. Perfect phylogenies may not exist. Other types of phylogenies such as near perfect phylogeny (Satya et al. (2006)) and imperfect phylogeny (Sridhar et al. (2006)) have been proposed for such kind of data.

More complicated structures, such as ancestral recombinant graph, can also been inferred and represent the evolutionary history among a set of samples.

Once these phylogenies (i.e., perfect/imperfect/near perfect or ancestral graph) are inferred for the various datasets, statistical models must be properly selected for examining the correlations in order to ensure the significance of any findings. For example, each partition indicated by an imperfect phylogeny should have different weight/confidence. While in the ancestral graph, the definition of partition also needs to be revised accordingly.

Another common challenge in analyzing the biological data is to deal with the existence of missing data and noise. There are two potential ways to tackle the problem: 1) Develop noise-tolerant association mapping methods which can automatically detect and remove outliers caused by missing/error data in the analysis; 2) Develop methods that integrate association mapping with inference of missing data under reliable statistical models such as maximum likelihood or maximum parsimony.

7.1.2 Association Mapping on Heterogenous Biological Data

Given the various biological data, combining the results can improve the robustness of the analysis. For example, if an association between a marker and a phenotype is repeatedly observed in different datasets, the association should be significant. There are two possible ways to combine the analysis:

- Post-process: Each dataset is analyzed independently. A post-processing algorithm is then used to analyze all the results, e.g., identifying repeated associations.
- Simultaneous-process: The analysis is integrated. An algorithm is used to detect associations in the multiple datasets simultaneously.

For example, in eQTL (expression quantitative trait loci) analysis, the SNP data (i.e., the genetic marker data) and the gene expression data are used together to detect the regulation between nucleotides and genes. More complex association models are required to analyze SNP data and gene expression data simultaneously. For example, in eQTL, the association may be between: 1) a single marker and a single gene expression; 2) a set of markers and a single gene expression; 3) a set of markers and a set of gene expressions.

The complex models and larger number of datasets pose a bigger computational challenge to association mapping analysis. Feasible solutions to the problems can be developed by either using approximation algorithms or incorporating highly efficient heuristics.

7.1.3 Complex Correlation Detection

The TreeQA and TreeQA+ algorithm are developed for the purpose of genome-wide association mapping. The idea is then extended in the TreeNL algorithm. The TreeNL algorithm is applied to the correlation clustering problem. As demonstrated in the experiments, TreeNL is able to find linearly correlated features effectively. However, even though a few non-linear correlations have been detected by TreeNL in the experiments, the non-linear correlations are hard to be detected in general.

In fact, there has been evidence showing that it is common in the high-dimensional data that the features have non-linear correlations. The analysis is more prone to spurious findings than in the case of linear correlations for the following two reasons:

- More data objects are required to confirm the existence of a complex correlation.
- Non-linear correlations are hard to be described and examined in regular statistical models.

Besides improving TreeNL by modifying its statistical model and test, other possible ways to improve non-linear correlation mining include:

1. Detect correlations/associations supported by a large amount of data objects with high density.
2. Allow users/experts to define a target model and only detect correlations/associations which follow the model.

The first approach is easier to implement but may sacrifice some detection power. The second approach provides more flexibility for complex correlation/association mining in different applications. But it requires more follow-up studies to solve key problems such as how to define a target model and how to use the model to effectively prune the searching space.

7.2 Maximum-diversity Sample Selection

In the diversity cover problem discussed in Chapter 5, each marker (i.e., feature) is given an equal weight. The problem can be extended to allowing a weight to be associated with each marker. The weight can be assigned to reflect the importance of each marker and may be dynamically adjusted. The weight of each uncovered marker is 1 before any sample is selected, and is assigned to the lowest dissimilarity of this marker to any covered marker[1]. The goal of this weighted diversity cover problem is to select samples such that the total weight of all markers is maximized.

[1]A marker is weighted 0 if it is identical to a covered marker and is weighted 1 if it is completely independent of any covered marker.

Also we can continue to investigate and evaluate alternative approaches that may offer further performance gains. An alternative greedy strategy for $PGDS$ is to start from the full set of samples and remove the sample that minimizes the decrease in diversity in each subsequent step until no sample can be further removed without violating the minimal diversity requirement. Note that a similar strategy can also be employed in the $K\rho DS$ algorithm which enumerates the sample subsets that can be removed without losing more than $1 - \rho$ diversity. In some cases where a minimum subset of diversity coverage ρ contains more than half of the samples, these alternative strategies can have a better runtime performance because they require a smaller search space.

The REP algorithm presented in Chapter 6 utilizes "mutual information" and "relative entropy" to selected maximum-diversity samples in the non-biallelic cases. Actually, there are other alternative measurements such as "total information" and "Pearson correlation". These alternative measurements may improve the effectiveness of the REP algorithm for datasets in different applications. And it can be interesting to compare the performance of these different measurements on different datasets.

Though the greedy solutions found by REP approximate the optimal solutions very well according to the experiments, the optimal solution is still preferred in many cases. If we can find a tight lower bound and upper bound properties of the mutual information measurements (or other alterative measurements), an exhaustive enumerating procedure similar to the $K\rho DS$ algorithm should also be feasible in the non-biallelic datasets.

In conclusion, in my thesis, I developed algorithms for important genetic analysis tasks such as genome-wide association mapping and maximum-diversity sample selection. As demonstrated by the extensive experiments, my algorithms are both efficient and effective in analyzing the genomic data.

Bibliography

Achtert, E., Böhm, C., Kriegel, H.-P., Kröger, P., and Zimek, A. (2006). Deriving quantitative models for correlation clusters. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 4–13. 57, 61

Agarwala, R., Fernandez-Baca, D., and Slutzki, G. (1995). Fast algorithms for inferring evolutionary trees. *Journal of Computational Biology*, 2(3):397–408. 6, 19, 22

Aggarwal, C. C., Wolf, J. L., Yu, P. S., Procopiuc, C., and Park, J. S. (1999). Fast algorithms for projected clustering. *SIGMOD Rec.*, 28(2):61–72. 62

Aggarwal, C. C. and Yu, P. S. (2000). Finding generalized projected clusters in high dimensional spaces. *SIGMOD Rec.*, 29(2):70–81. 57, 58, 61

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 94–105. 62

Akey, J., Jin, L., and Xiong, M. (2001). Haplotypes vs single marker linkage disequilibrium tests: what do we gain? *Eur J. Hum Genet.*, 9(4):291–300. 2, 3, 14, 16, 29, 49, 142

Andritsos, P., Tsaparas, P., Miller, R. J., and Sevcik, K. C. (2003). Limbo: Scalable clustering of categorical data. *Hellenic Database Symposium*. 123, 135

Basu, S., Bilenko, M., and Mooney, R. J. (2004). A probabilistic framework for semi-supervised clustering. *KDD'04 Proceedings*. 124

Böhm, C., Kailing, K., Kröger, P., and Zimek, A. (2004). Computing clusters of correlation connected objects. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 455–466. 57, 58, 60, 61

Chakrabarti, D., Papadimitriou, S., Modha, D., and Faloutsos, C. (2004). Fully automatic cross-associations. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 97

Churchill, G. A., Airey, D. C., Allayee, H., Angel, J. M., Attie, A. D., Beatty, J., Beavis, W. D., Belknap, J. K., Bennett, B., Berrettini, W., Bleich, A., Bogue, M., Broman, K. W., Buck, K. J., Buckler, E., Burmeister, M., Chesler, E. J., Cheverud, J. M., Clapcote, S., Cook, M. N., Cox, R. D., Crabbe, J. C., Crusio, W. E., Darvasi, A., Deschepper, C. F., Doerge, R. W., Farber, C. R., Forejt, J., Gaile, D., Garlow, S. J., Geiger, H., Gershenfeld, H., Gordon, T., Gu, J., Gu, W., de Haan, G., Hayes, N. L., Heller, C., Himmelbauer, H., Hitzemann, R., Hunter, K., Hsu, H.-C., Iraqi, F. A., Ivandic, B., Jacob, H. J., Jansen, R. C., Jepsen, K. J., Johnson, D. K., Johnson, T. E., Kempermann, G., Kendziorski, C., Kotb, M., Kooy, R. F., Llamas, B., Lammert, F., Lassalle, J.-M., Lowenstein, P. R., Lu, L., Lusis, A., Manly, K. F., Marcucio, R., Matthews, D., Medrano, J. F., Miller, D. R., Mittleman, G., Mock, B. A., Mogil, J. S., Montagutelli, X., Morahan, G., Morris, D. G., Mott, R., Nadeau, J. H., Nagase, H., Nowakowski, R. S., O'Hara, B. F., Osadchuk, A. V., Page, G. P., Paigen, B., Paigen, K., Palmer, A. A., Pan, H.-J., Peltonen-Palotie, L., Peirce, J., Pomp, D., Pravenec, M., Prows, D. R., Qi, Z., Reeves, R. H., Roder, J., Rosen, G. D., Schadt, E. E., Schalkwyk, L. C., Seltzer, Z., Shimomura, K., Shou, S., Sillanp, M. J., Siracusa, L. D., Snoeck, H.-W., Spearow, J. L., Svenson, K., Tarantino, L. M., Threadgill, D., Toth, L. A., Valdar, W., de Villena, F. P.-M., Warden, C., Whatley, S., Williams, R. W., Wiltshire, T., Yi, N., Zhang, D., Zhang, M., and Zou, F. (2004). Complex trait consortium. the collaborative cross, a community resource for the genetic analysis of complex traits. *NATURE GENETICS*, 36(11):1133–7. 3, 118

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). Introduction to algorithms, second edition. *MIT Press and McGraw-Hill*. 9, 10, 16, 25, 99, 103, 105

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. Wiley, New York. 123

Dash, M. and Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis*, 1(3). 97

Dhillon, I. S., Mallela, S., and Kumar, R. (2003a). A divisive information-theoretic feature clustering algorithm for text classification. *Jounal of Machine Learning Research*, (3):1265–1287. 124

Dhillon, I. S., Mallela, S., and Modha, D. S. (2003b). Information-theoretic co-clustering. In *SIGKDD '03 Conference Proceedings*. 97

Ding, Z., Mailund, T., and Song, Y. S. (2008). Efficient whole-genome association mapping using local phylogenies for unphased genotype data. *Bioinformatics*, 24(19):2215–2221. 143

Edwards, A. W. F. and Cavalli-Sforza, L. L. (1964). Reconstruction of evolutionary trees. In *V. H. Heywood and J. McNeill, Phenetic and phylogenetic classification*, 6. 38

Felsenstein, J. (1981). Evolutionary trees from gene frequencies and quantitative characters: Finding maximum likelihood estimates. *Evolution*, 35(6):1229–1242. 38, 40, 41, 42, 48, 52

Fernandez-Baca, D. (2001). *The Perfect Phylogeny Problem Steiner Trees in Industry*. Kluwer Academic Press. 6

Fisher, R. A. (1935). *The Design of Experiment*. Hafner, New York. 2

Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(3):133–151. 113

Guiasu, S. (1977). *Information Theory with Applications*. McGraw-Hill, New York. 9, 12

Hartmann, K. and Steel, M. (2006). Maximimizing phylogenetic diversity in biodiverstity conservation: greedy solutions to the noah's ark problem. *Systematic Biology*, 55(4):644–651. 95

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer-Verlag, New York. 123

Hinneburg, A. and Keim, D. A. (1999). Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 506–517. 62

Hochbaum, D. S. and Pathria, A. (1998). Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Quarterly*, 45:615–627. 97

Hudson, R. R. and Kaplan, N. L. (1985). Statistical properties of the number of recombination events in the history of a sample of dna sequences. *Genetics*, 111(1):147–164. 6, 15, 17, 38

Ideraabdullah, F., la Casa-Esperon, E. D., Bell, T., Detwiler, D., Magnuson, T., and C. Sapienza, F. P.-M. d. V. (2004). Genetic and haplotype diversity among wild derived mouse inbred strains. *Genome Res*, 14:1880–1887. 5, 95

Jin, C., Lan, H., Attie, A. D., Churchill, G. A., Bulutuglo, D., and Yandell, B. S. (2004). Selective phenotyping for increased efficiency in genetic mapping studies. *Genetics*, 168(4):2285–93. 95

Kannan, R., Vempala, S., and Veta, A. (2000). On clusterings: good, bad, and spectral. In *IEEE Annual Symposium on Foundations of Computer Science*. 123

Kumar, R., Mahadevan, U., and D.Sivakumar (2004). A graph-theoretic approach to extract storylines from search results. *KDD'04 Proceedings*. 124

Larribea, F., Lessarda, S., and Schork, N. J. (2002). Gene mapping via the ancestral recombination graph. *Theoretical Population Biology*, 62(2):215–229. 17, 36

Liu, J. and Wang, W. (2003). Op-cluster: Clustering by tendency in high dimensional space. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 187. 62

Loughry, J., Hemert, J. I. V., and Schoofs, L. (2002). *Efficiently Enumerating the Subsets of a Set*. 47

Mailund, T., Besenbacher, S., and Schierup, M. H. (2006). Whole genome association mapping by incompatibilities and local perfect phylogenies. *BMC Bioinformatics*, 7(454). 2, 3, 6, 15, 17, 29, 36, 49, 141

Mailund, T., Schierup, M. H., Pedersen, C. N., Mechlenborg, P. J., Madsen, J. N., and Schauser, L. (2005). Coasim: A flexible environment for simulating genetic data under coalescent model. *BMC Bioinformatics*, 6(252). 29, 49

McClurg, P., Pletcher, M. T., Wiltshire, T., and Su, A. I. (2006). Comparative analysis of haplotype association mapping algorithms. *BMC Bioinformatics*, 7(61). 2, 3, 16, 29, 49, 142

Miller, R. G. (1981). *Simultaneous Statistical Inference*. Springer Verlag, New York. 15

Minichiello, M. J. and Durbin, R. (2006). Mapping trait loci by use of inferred ancestral recombination graphs. *Am J Hum Genet*, 79(5):910–922. 17, 36

Morris, A. P., Whittaker, J. C., and Balding, D. J. (2002). Fine-scale mapping of disease loci via shattered coalescent modeling of genealogies. *Am J Hum Genet*, 70(3). 17, 36

Nelson, E. (1967). *Dynamical theories of Brownian motion*. 38, 40, 41, 43, 52

Onkamo, P., Ollikainen, V., Sevon, P., and *et al* (2002). Association analysis for quantitative traits by data mining: Qhpm. *Ann. Hum. Genet.*, 66:419–429. 2, 3, 16, 29, 49, 142

Pan, F., McMillan, L., de Villena, F. P., Threadgill, D., and Wang, W. (2009). Treeqa: Quantitative genome wide association mapping using local perfect phylogeny trees. In *Proceedings of The 14th Pacific Symposium on Biocomputing (PSB)*. 6, 39, 49

Pan, F., Roberts, A., McMillan, L., de Villena, F. P., Threadgill, D., and Wang, W. (2007). Sample selection for maximal diversity. In *Proceedings of The Internaltionl Conderence on Data Mining (ICDM)*. 10

Pan, F., Wang, W., Tung, A. K., and Yang, J. (2005a). Finding representative sets from massive data. *IEEE International Conference on Data Mining*. 12

Pan, F., Wang, W., Tung, A. K. H., and Yang, J. (2005b). Finding representative set from massive data. *Technical Report: TR05-014*. 137

Pe'er, I., de Bakker, P. I. W., Maller, J., Yelensky, R., Altshuler, D., and Daly, M. J. (2006). Evaluating and improving power in whole-genome association studies using fixed marker sets. *Nature Genetics*, 38:663–667. 2, 3, 14, 16, 29, 49, 142

Procopiuc, C. M., Jones, M., Agarwal, P. K., and Murali, T. M. (2002). A monte carlo algorithm for fast projective clustering. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 418–427. ACM. 62

Roberts, A., McMillan, L., Wang, W., Parker, J., Rusyn, I., and Threadgill, D. (2007). Inferring missing genotypes in large snp panels using fast nearest-neighbor searches over sliding windows. *ISMB*. 113

Satya, R. V., Mukherjee, A., Alexe, G., Parida, L., and Bhanot, G. (2006). Constructing near-perfect phylogenies with multiple homoplasy events. *Bioinformatics*, 22(14):e514–e522. 143, 144

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464. 69

Sevon, P., Toivonen, H., and Ollikainen, V. (2006). Tree pattern mining for gene mapping. *IEEE Transactions on Computational Biology and Bioinformatics*, 3(2). 2, 3, 6, 15, 17, 29, 36, 49, 141

S.Hochbaum, D. and Pathria, A. (1998). Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Quarterly*, (45):615–627. 122, 124, 132

Slonim, N. and Tishby, N. (2000). Document clustering using word clusters via the information bottleneck method. *ACM SIGIR 2000*. 123, 137

Sridhar, S., Blelloch, G., Ravi, R., and Schwartz, R. (2006). Optimal imperfect phylogeny reconstruction and haplotyping. In *Proceedings of Comput Syst Bioinformatics Conf.*, pages 199–210. 143, 144

Steel, M. (2005). Phylogenetic diversity and the greedy algorithm. *Syst. Biol*, 54:527–529. 95

Szatkiewicz, J. P., Beane, G. L., Ding, Y., Hutchins, L., de Villena, F. P. M., and Churchill, G. A. (2008). An imputed genotype resource for the laboratory mouse. *Mammalian Genome*, 19(3). 32, 51

Threadgill, D. W., Hunter, K. W., and Williams, R. W. (2002). Genetic dissection of complex and quantitative traits: from fantasy to reality via a community effort. *Mamm Genome*, 13:175–178. 3

Toivonen, H. T. T., Onkamo, P., Vasko, K., Ollikainen, V., Sevon, P., Mannila, H., Herr, M., and Kere, J. (2000). Data mining applied to linkage disequilibrium mapping. *Am J Hum Genet*, 67:133–145. 15

Tung, A. K. H., Xu, X., and Ooi, B. C. (2005). Curler: finding and visualizing nonlinear correlation clusters. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 467–478. 58, 60, 61, 80, 81, 82, 93

Waldron, E., Whittaker, J., and Balding, D. (2005). Fine mapping of disease genes via haplotype clustering. *Genetic Epidemiology*, 30:170–179. 15

Wang, J. and Karypis, G. (2004). Summary: Efficiently summarizing transactions for clustering. *ICDM'04 Proceedings*. 134

Wang, X. and Paigen, B. (2002). Quantitative trait loci and candidate genes regulating hdl cholesterol. *Arteriosclerosis, Thrombosis, and Vascular Biology*, 22(1390). 32, 33, 51

Wang, X. and Paigen, B. (2005). Haplotype-based linkage disequilibrium mapping via direct data mining. *Bioinformatics*, 21(24):4384–4393. 2, 3, 15, 17, 29, 49, 142

Williams, R. W., Gu, J., Qi, S., and Lu, L. (2001). The genetic structure of recombinant inbred mice: high-resolution consensus maps for complex trait analysis. *Genome Biol*, 2(11):RESEARCH0046. 95

Xu, Z., Zou, F., and Vision, T. J. (2005). Improving quantitative trait loci mapping resolution in experimental crosses by the use of genotypically selected samples. *Genetics*, 170(1):401–8. 95

Yang, H., Bell, T. A., Churchill, G. A., and de Villena, F. P. M. (2007). On the subspecific origin of the laboratory mouse. *Nature Genetics*, 39. 32, 51

Zhang, X., Pan, F., and Wang, W. (2008). Care: Finding local linear correlations in high dimensional data. In *ICDE '08: Proceedings of the 24th International Conference on Data Engineering*, pages 130–139. 57, 58, 59, 61, 80, 81, 88, 93

Zollner, S. and Pritchard, J. K. (2005). Coalescent-based association mapping and fine mapping of complex trait loci. *Genetics*, 169(2):1071–92. 2, 3, 6, 8, 15, 17, 29, 36, 49, 141