Efficient Algorithms for Detecting Genetic Interactions in Genome-Wide Association Study

Xiang Zhang

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
2011

Approved by:

Wei Wang, Advisor

Fei Zou, Reader

Leonard McMillan, Reader

Jan F. Prins, Reader

David W. Threadgill, Reader

Abstract

Xiang Zhang: Efficient Algorithms for Detecting Genetic Interactions in Genome-Wide Association Study.
(Under the direction of Wei Wang.)

Genome-wide association study (GWAS) aims to find genetic factors underlying complex phenotypic traits, for which epistasis or gene-gene interaction detection is often preferred over a single-locus approach. However, the computational burden has been a major hurdle to apply epistasis test at the genome-wide scale due to the large number of single nucleotide polymorphism (SNP) pairs to be tested. We have developed and implemented a series of efficient algorithms, i.e., FastANOVA, FastChi, COE, and TEAM, that support epistasis tests in a wide range of problem settings. These algorithms utilize a permutation test for proper error control. Unlike heuristic approaches, they guarantee to find the optimal solutions. It has been shown theoretically and experimentally that these algorithms significantly speed up the process of epistasis detection.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Genome-wide association study (GWAS) examines the genetic variants across the entire genome to identify genetic factors associated with observed phenotypes. It has been shown to be a promising design to locate the genetic factors causing phenotypic differences (Saxena et al. (2007); The Wellcome Trust Case Control Consortium (2007)). Since most traits of interest are complex, finding gene-gene interaction has received increasing attention in recent years (Cordell (2009); Musani et al. (2007)).

1.1 Genome-Wide Association Study

The most abundant source of genetic variations are single nucleotide polymorphisms (SNPs). A SNP is a DNA sequence variation occurring when a single nucleotide (A, T, G, or C) in the genome differs between individuals of a species. For inbred species, such as inbred mice, a SNP usually shows variation between only two of the four possible nucleotide types (Ideraabdullah et al. (2004)), which allows us to represent it by a binary variable. The binary representation of a SNP is also referred to as the *genotype* of the SNP. Recent advances in high-throughput techniques enable genotyping SNPs in genome-wide scale, resulting in large datasets containing thousands to millions of SNPs, e.g. the genotype datasets available in the Broad Institute ($http://www.broad.mit.edu/$) and the Jackson Laboratory ($http://www.jax.org/$).

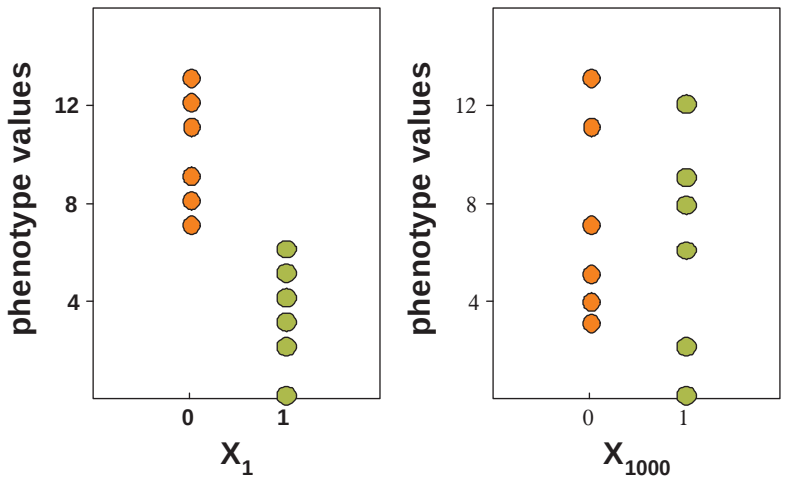
SNPs							Phenotype
X_1	X_2	X_3	X_4	X_5	$\cdots$	X_{1000}	Y
0	0	0	1	0		1	8
0	0	0	0	0		0	7
0	1	1	0	0	$\cdots$	1	12
0	1	0	0	1		0	11
0	1	0	1	0		1	9
0	1	0	0	0	$\cdots$	0	13
1	0	1	1	1		1	6
1	0	0	0	1		0	4
1	1	1	1	1	$\cdots$	1	2
1	0	0	1	0		0	5
1	0	0	1	0		1	0
1	0	1	1	0	$\cdots$	0	3

Table 1.1: An example dataset in genome-wide association study

A phenotype is an observable trait or characteristic of an individual. Phenotypes can be either quantitative or binary. Examples of quantitative phenotypes are height and weight. These phenotypes can be represented by continuous variables. Binary phenotypes are usually studied in case-control studies. In such studies, the samples either have or do not have a certain disease. We can use $\{0,1\}$ to indicate the disease status of an individual. Table 1.1 shows an example dataset consisting of 1000 SNPs $\{X_1, X_2, \cdots, X_{1000}\}$ and a quantitative phenotype Y for 12 individuals.

Genome-wide association studies (GWAS) find associations between SNPs and phenotypes across a set of individuals under study. More formally, let $X = \{X_1, X_2, \cdots, X_N\}$ be the set of N SNPs for M individuals in the study, and Y be the phenotype of interest. The goal of GWAS is to find SNPs in X, that are highly associated with Y.

Various statistics, such as ANOVA (analysis of variance) test and chi-square test, can be applied to measure the association between SNPs and the phenotypes of interest. Here, we take ANOVA test as an example. ANOVA test is one of the standard statistical methods routinely used in quantitative phenotype association study (Pagano and Gauvreau (2000)). The goal of ANOVA test is to determine whether the group means are significantly different after



(a) Strong association (b) No association

Figure 1.1: Examples of associations between a phenotype and two different SNPs

accounting for the variances within groups. It accomplishes the comparison by decomposing the total variance in the data into within-group variance and between-group variance. If the between-group variance is sufficiently larger than the within-group variance, then the test concludes that there is significant (phenotypic) difference between the groups. In the application of genetic association study, the individuals' phenotype values are grouped by the genotype of a SNP or a subset of SNPs. Using the dataset showing in Table 1.1, Figure 1.1(a) shows an example of strong association between the phenotype and SNP X_1. 0 and 1 on the x-axis represent the binary SNP genotype and the y-axis represents the phenotype. Each point in the figure represents an individual. It is clear from the figure that the phenotype values are partitioned into two groups with distinct means, hence indicating a strong association between the phenotype and the SNP. On the other hand, if the genotype of a SNP partitions the phenotype values into groups as shown in Figure 1.1(b), the phenotype and the SNP are not associated with each other.

1.2 Epistasis Detection and Challenges

Many phenotypes of interest are complex in the sense that they are likely caused by the joint effects of multiple genes (Carlson et al. (2004); Segr et al. (2005)). In order to understand the underlying biological mechanisms of complex phenotype, one needs to consider the joint effect of multiple SNPs simultaneously. The interaction between genes is also referred to as *epistasis* (Cordell (2009)). Although the idea of studying the association between phenotype and multiple SNPs is straightforward, the implementation is nontrivial. For a study with total N SNPs, in order to find the association between n SNPs and the phenotype, a brute-force approach is to exhaustively enumerate all $\binom{N}{n}$ possible SNP combinations and evaluate their associations with the phenotype. The computational burden imposed by this enormous search space often makes the complete genome-wide association study intractable.

The computational challenge of genome-wide association study is further compounded by another well-known statistical problem – the multiple testing problem (Miller (1981)). The multiple testing problem can be described as the potential increase in Type I error when statistical tests are performed multiple times. Let α be the Type I error for each independent test. If n independent comparisons are performed, the experimental-wise error α' is given by

$$\alpha' = 1 - (1 - \alpha)^n.$$

For example, when $\alpha = 0.05$ and $n = 20$, $\alpha' = 1 - 0.95^{20} = 0.64$. We have 64% probability to get at least one spurious result. Determining the statistical significance of the association between the phenotype and SNPs is crucial. Bonferroni correction based on the assumption that all n tests are independent is too conservative for the genome-wise association studies since SNPs are often correlated. Alternatively, a permutation procedure can be used and it is much preferred in association studies which automatically takes the correlation structure of SNPs into consideration.

The null hypothesis is that there is no association between the genotype and the phenotype.

Permutation test is used to estimate the null distribution (Churchill and Doerge (1994)). The idea is to randomly permute the phenotype K times, where K can be hundreds to thousands. The association analysis will be repeated in order to find the maximum test value for each permutated phenotype. Then the distribution of the K maximum test values is used as the approximated null distribution to assess the statistical significance of the findings from the original phenotype. Permutation test is usually very time-consuming since the test procedure needs to be performed in all permutations in order to find the maximum values.

Algorithm development to support these large scale analysis is still in its early stage. Most existing work focuses on studying associations between the phenotype and SNP-pairs and can only handle a small number of SNPs. Given a pair of SNPs, the phenotype values can be partitioned into at most four groups by the genotype of the SNP-pair, i.e., 00, 01, 10, and 11. Since each SNP has a distinct location on the genome, the association study of a phenotype and SNP-pairs is also called *two-locus association mapping*. Important findings are appearing in the literature from studying the association between phenotypes and SNP-pairs (Saxena et al. (2007); Scuteri et al. (2007); Weedon et al. (2007)).

Although various statistical tests have been routinely applied to find association between SNP-pairs and phenotype, they are usually not performed in genome-wide scale. This is due to the fact that the search space of two-locus association mapping in genome-wide scale prohibits an exhaustive search. Suppose that the dataset consists of N SNPs and the number of permutations is K. The total number of tests is $KN(N-1)/2$. Given a moderate number of SNPs $N = 10,000$ and number of permutations $K = 1,000$, the number of tests is around 5×10^{10}. Efficient algorithms are needed to enable epistasis detection in the whole-genome scale.

1.3 Thesis Statement

Efficient exhaustive algorithms can be designed for two-locus epistasis detection in genome-wide association study. The proposed algorithms incorporate large permutation test for error controlling. They guarantee to find the optimal solution. By applying effective pruning strategies, the computational cost of these algorithms can be dramatically reduced. Extensive experimental results demonstrate that the proposed algorithms are orders of magnitude faster than brute force alternatives.

1.4 Overview of the Developed Algorithms

This thesis presents a set of algorithms for two-locus epistasis detection. These programs use the permutation procedure for proper error control. They are exhaustive and accurate in the sense that no significant epistatic interactions between SNP-pairs are skipped. It has been theoretically proved and experimentally validated that these algorithms greatly speed up the epistasis test process. We give a brief overview of the designing principles of these programs here. All the algorithms utilize search space pruning to reduce the computational cost of epistatic test.

The FastANOVA (Zhang et al. (2008)) algorithm is designed for ANOVA test. It utilizes an upper bound of the two-locus ANOVA test to prune the search space. The upper bound is expressed as the sum of two terms. The first term is based on the single-SNP ANOVA test. The second term is based on the genotype of the SNP-pair and is independent of permutations. This property allows to index SNP-pairs in a 2D array based on the genotype relationship between SNPs. Since the number of entries in the 2D array is bounded by the number of individuals in the study, many SNP-pairs share a common entry. Moreover, it can be shown that all SNP-pairs indexed by the same entry have exactly the same upper bound. Therefore, we can compute the upper bound for a group of SNP-pairs together. Another important property is that the indexing structure only needs to be built once and can be reused

for all permutated data. Utilizing the upper bound and the indexing structure, FastANOVA only needs to perform the ANOVA test on a small number of candidate SNP-pairs without the risk of missing any significant pair.

The principal used in FastANOVA can also be applied to chi-square test. We can develop an upper bound for chi-square test, which is also expressed as the sum of two terms. The first term is based on the single-SNP chi-square test. The second term is based on the genotype of the SNP-pair and is independent of permutations. Based on this observation, we developed the FastChi algorithm (Zhang et al. (2009)).

The COE algorithm (Zhang et al. (2010)) takes the advantage of convex optimization. It can be shown that a wide range of statistical tests, such as chi-square test, likelihood ratio test (also known as G-test), and entropy-based tests are all convex functions of observed frequencies in contingency tables. Since the maximum value of a convex function is attained at the vertices of its convex domain, by constraining on the observed frequencies in the contingency tables, we can determine the domain of the convex function and get its maximum value. This maximum value is used as the upper bound on the test statistics to filter out insignificant SNP-pairs. COE is applicable to all tests that are convex.

FastANOVA, FastChi, and COE are designed for studies with homozygous genotypes and relatively small sample sizes. In human GWAS, heterozygous genotypes are common, and the number of individuals can be large. We therefore developed the third program, TEAM, that is suitable for human GWAS. The basic idea of TEAM is that it incrementally updates the contingency tables of two-locus test by utilizing a minimum spanning tree. The nodes of the tree are SNPs and the edges represent the difference between two connected SNPs. It can be shown that we can get the exact test values by searching the minimum spanning tree without scanning all individuals. TEAM records the test statistics of all SNP-pairs instead of just the ones with high values. Thus it allows family-wise error rate (FWER) and false discovery rate (FDR) calculation.

1.5 Thesis Outline

The thesis is organized as follows:

- The FastANOVA algorithm is presented in Chapter 2.
- The FastChi algorithm is presented in Chapter 3.
- The COE algorithm is presented in Chapter 4.
- The TEAM algorithm is presented in Chapter 5.
- Chapter 6 concludes the thesis work.

Chapter 2

The FastANOVA Algorithm

2.1 Introduction

Quantitative phenotype association study analyzes genetic variation across a population in order to find the genetic factors underlying continuous phenotypes (such as height or weight). ANOVA (analysis of variance) test is one of the standard statistic methods and has been routinely used in quantitative phenotype association study (Pagano and Gauvreau (2000)). ANOVA test is used to determine whether the group means are significantly different. The total variance in the data is divided into within-group variance and between-group variance. If the between-group variance is sufficiently larger than the within-group variance, then the test concludes that there is significant phenotypic difference between the groups. Although ANOVA test has been a valuable tool to find association between SNP-pairs and quantitative phenotype, it is usually not performed at a genome-wide scale due to the enormous search space.

In this chapter, we examine the *computational aspect* of ANOVA test. We present an efficient algorithm, FastANOVA, and show that the standard ANOVA test can be applied in genome-wide scale for two-locus association mapping even when the permutation procedure is needed. Unlike algorithms applying heuristics, FastANOVA is a *complete* algorithm, i.e., it guarantees to find the optimal solution, though it does not explicitly examine all possible

SNP-pairs. In fact, a large portion of the SNP-pairs are pruned without the need of performing the tests. FastANOVA establishes an upper bound on the two-locus ANOVA test. The upper bound is the sum of two terms: one based on the ANOVA test between phenotype and a single SNP, and the other based on the pair-wise SNP genotype and the ordered phenotype values. This formulation of the upper bound allows the algorithm to calculate the bound for a large number of SNPs together, which enables fast candidate retrieval. Moreover, the intermediate results for calculating the second term of the upper bound is independent of phenotype permutations. Hence they only need to be computed once and can be reused in all permutations. Applying this bound, FastANOVA is able to identify SNP-pairs with significant ANOVA test values using only a small fraction of the time required by performing ANOVA test on all SNP-pairs. The principles developed in FastANOVA are also applicable to the other statistical tests such as Chi-square test which is commonly used in case-control study where phenotypes are binary variables.

2.2 Related Work

The problem of genetic association study has attracted extensive research interests. In this section, we review the related work from a computational point of view. Please refer to (Doerge (2002); Hoh and Ott (2003); Balding (2006)) for excellent surveys of existing work.

Different machine learning models have been adopted in multilocus association study. In (Curtis et al. (2001); Sherriff and Ott (2001)), the authors investigate using neural networks to study the relationship between complex traits and multilocus genotypes. These models are theoretically well suited for analyzing high-order interactions. However, the results of these methods are usually expressed as weights associated with SNPs. They are difficult to interpret and do not clearly identify the interacting SNPs. Recursive partitioning methods (Zhang and Bonney (2000); Province et al. (2001)) utilize classification and regression tree (CART) (Breiman et al. (1984)) to pick the SNP that minimizes some pre-specified measure

of impurity in each iteration. These methods are not effective in detecting SNP combinations if there is little or no marginal effect.

Under the assumption that the number of SNPs is limited, e.g., from tens to hundreds, exhaustive algorithms that explicitly enumerate all possible SNP combinations have been developed. Combinatorial partitioning method (CPM) (Nelson et al. (2001)) is designed to identify multilocus genotypic partitions that predict quantitative trait variation. Given a small set of SNPs, CPM searches for the partitions of multilocus genotypes that are the most predictive in terms of phenotypic variability. Motivated by CPM, multifactorial dimension reduction (MDR) (Ritchie et al. (2001); Moore et al. (2006)) is designed for case/control studies. By pooling genotypes of multilocus into two groups at high disease risk and low disease risk, MDR reduces the genotype of multiple SNPs into one dimension. Among all possible combinations, MDR selects the one that maximizes the case/control ratio of the high risk group. Since these methods explicitly enumerate all possible SNP combinations, they are not well adapted to genome-wide association studies.

To avoid exhaustively enumerating the search space, a common approach is to break the problem into two steps (Hoh et al. (2000); Evans et al. (2006)). First, a subset of important SNPs are selected. Second, within the selected subset, the association between SNPs and the phenotypes are searched. These methods are not complete since the SNPs with weak marginal effects may not be selected in the first step. Genetic algorithm (Carlborg et al. (2000); Nakamichi et al. (2001)) has been applied in finding SNP-pairs for quantitative phenotypes. These methods cannot guarantee to find the optimal solution.

Feature selection methods (Liu and Motoda (1998)) have been proposed to address the problem of finding important SNPs. In feature selection, the selected feature subset usually contains features that have low correlation with each other but have strong correlation with the target feature. In the application of selecting SNPs, the goal is to select a subset of SNPs that can be used as proxies for all SNPs in the genome (Sebastiani et al. (2003); Chi et al. (2006); Halperin et al. (2005)). The selected SNPs can then be used as the input SNPs in the

(a) Grouping of Y by X_i

$X_i = 1$	$X_i = 0$
group A	group B

(b) Grouping of Y by X_iX_j

	$X_i = 1$	$X_i = 0$
$X_j = 1$	group a_1	group b_1
$X_j = 0$	group a_2	group b_2

Table 2.1: Possible groupings of phenotype values by the genotypes of X_i and (X_iX_j)

association study. These methods are also not complete since some important SNPs may not be tagged.

2.3 The Problem

In this section, we formalize the problem of two-locus ANOVA test with permutation procedure. Let $\{X_1, X_2, \cdots, X_N\}$ be the set of SNPs of M individuals ($X_i \in \{0, 1\}, 1 \leq i \leq N$) and $Y = \{y_1, y_2, \cdots, y_M\}$ be the quantitative phenotype of interest, where y_m ($1 \leq m \leq M$) is the phenotype value of individual m.

For any SNP X_i ($1 \leq i \leq N$), we represent the F-statistic from the ANOVA test of X_i and Y as $F(X_i, Y)$. For any SNP-pair (X_iX_j), we represent the F-statistic from the ANOVA test of (X_iX_j) and Y as $F(X_iX_j, Y)$.

The basic idea of ANOVA test is to partition the total sum of squared deviations SS_T into between-group sum of squared deviations SS_B and within-group sum of squared deviations SS_W:

$$SS_T = SS_B + SS_W.$$

In the application of two-locus association study, Table 3.0(a) and Table 3.0(b) show the possible groupings of phenotype values by the genotypes of X_i and (X_iX_j) respectively. Let A, B, a_1, a_2, b_1, b_2 represent the groups as indicated in Table 3.0(a) and Table 3.0(b). We use $SS_B(X_i, Y)$ and $SS_B(X_iX_j, Y)$ to distinct the one locus (i.e., single-SNP) and two locus

(i.e., SNP-pair) analyses. Specifically, we have

$$SS_T(X_i, Y) = SS_B(X_i, Y) + SS_W(X_i, Y),$$

$$SS_T(X_iX_j, Y) = SS_B(X_iX_j, Y) + SS_W(X_iX_j, Y).$$

The F-statistics for ANOVA tests on X_i and (X_iX_j) are:

$$F(X_i, Y) = \frac{M-2}{2-1} \times \frac{SS_B(X_i, Y)}{SS_T(X_i, Y) - SS_B(X_i, Y)}, \tag{2.1}$$

$$F(X_iX_j, Y) = \frac{M-g}{g-1} \times \frac{SS_B(X_iX_j, Y)}{SS_T(X_iX_j, Y) - SS_B(X_iX_j, Y)}, \tag{2.2}$$

where g in Equation (2.2) is the number of groups that the genotype of (X_iX_j) partitions the individuals into. Possible values of g are 3 or 4, assuming all SNPs are distinct: If none of groups A, B, a_1, a_2, b_1, b_2 is empty, then $g = 4$. If one of them is empty, then $g = 3$.

Let $T = \sum_{y_m \in Y} y_m$ be the sum of all phenotype values. The total sum of squared deviations does not depend on the groupings of individuals:

$$SS_T(X_i, Y) = SS_T(X_iX_j, Y) = \sum_{y_m \in Y} y_m^2 - \frac{T^2}{M}.$$

Let $T_{group} = \sum_{y_m \in group} y_m$ be the sum of phenotype values in a specific group, and n_{group} be the number of individuals in that group. $SS_B(X_i, Y)$ and $SS_B(X_iX_j, Y)$ can be calculated as follows:

$$SS_B(X_i, Y) = \frac{T_A^2}{n_A} + \frac{T_B^2}{n_B} - \frac{T^2}{M},$$

$$SS_B(X_iX_j, Y) = \frac{T_{a_1}^2}{n_{a_1}} + \frac{T_{a_2}^2}{n_{a_2}} + \frac{T_{b_1}^2}{n_{b_1}} + \frac{T_{b_2}^2}{n_{b_2}} - \frac{T^2}{M}.$$

Note that for any group of A, B, a_1, a_2, b_1, b_2, if $n_{group} = 0$, then $\frac{T_{group}^2}{n_{group}}$ is defined to be

0.

The two-locus association mapping with permutation test is typically conducted in the following way (Pagano and Gauvreau (2000); Pesarin (2001); Mielke and Berry (2001)).

First, for every SNP-pair (X_iX_j) $(1 \leq i < j \leq N)$, the ANOVA test is performed and $F(X_iX_j, Y)$ is recorded.

Second, a permutation test is performed to get a reference distribution in order to assess the statistical significance of previous findings. More specifically, a permutation Y_k of Y is generated by sampling the phenotype Y without replacement. In other words, phenotype values are randomly assigned to individuals in the dataset with no single phenotype value being assigned to more than one individual. Let $Y' = \{Y_1, Y_2, \cdots, Y_K\}$ be the set of K permutations of Y. For each permutation $Y_k \in Y'$, let F_{Y_k} represent the maximum F-statistic value of all SNP-pairs, i.e.,

$$F_{Y_k} = \max\{F(X_iX_j, Y_k) | 1 \leq i < j \leq N\}.$$

The distribution of $\{F_{Y_k} | Y_k \in Y'\}$ is then used as the reference distribution for assessing the statistical significance of $F(X_iX_j, Y)$ values found using the original phenotype Y: Given a Type I error threshold α, the *critical value* F_α is the αK-th largest value in $\{F_{Y_k} | Y_k \in Y'\}$. The SNP-pair (X_iX_j) whose F-statistic value $F(X_iX_j, Y) \geq F_\alpha$ is considered as significant at α.

For example, Figure 2.1 shows the cumulative distribution of the maximum values for $K = 100$ permutations. Suppose that $\alpha = 0.3$, then F_α is the 30th largest value among the 100 maximum test values, which is 32 as shown in this example.

Two computational problems need to be solved in this procedure. The first one is to find the critical value F_α for a given Type I error threshold α. The second one is to find all SNP-pairs (X_iX_j) whose F-statistics are greater than F_α. We formalize these two problems as follows.

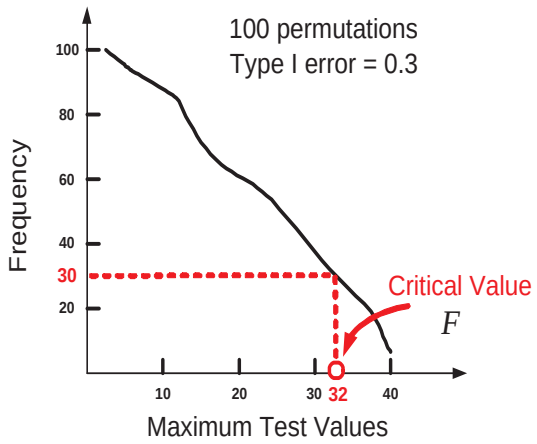


Figure 2.1: An example of determining the critical value using permutation test

Problem (1): Given the Type I error threshold α, find the critical value F_α, which is the αK-th largest value in $\{F_{Y_k} | Y_k \in Y'\}$.

Problem (2): Given the threshold F_α, find all significant SNP-pairs $(X_i X_j)$ such that $F(X_i X_j, Y) \geq F_\alpha$.

A brute force approach to these two problems is to enumerate all SNP-pairs and find their F-statistics. In Problem (1), for each permutation $Y_k \in Y$, all SNP-pairs need to be enumerated in order to find the maximum value F_{Y_k}. In Problem (2), all SNP-pairs need to be enumerated to see if their test values are above the threshold F_α. Computationally, Problem (1) is more challenging, since the permutation number K can range form hundreds to thousands, which means the running time of finding the critical value F_α can be hundreds to thousands times longer than the running time of finding the significant SNP-pairs in Problem (2) using a brute-force search.

In the reminder of this chapter, we first derive an upper bound on two-locus ANOVA test value and discuss how this upper bound enables an efficient ANOVA testing for a single phenotype. Then we show how this approach can be easily extended to handle the permutation procedure.

2.4 The Upper Bound

2.4.1 Updating F-Statistic

Since the total sum of squared deviations does not change, from the calculation of $F(X_i, Y)$ and $F(X_iX_j, Y)$ (Equations (2.1) and (2.2)), we know that the relationship between these two tests only depends on the relationship between $SS_B(X_i, Y)$ and $SS_B(X_iX_j, Y)$. Next we show that $SS_B(X_iX_j, Y)$ can be updated from $SS_B(X_i, Y)$.

For groups A, a_1 and a_2, let

$$\begin{aligned}
\Delta A &= \frac{T_{a_1}^2}{n_{a_1}} + \frac{T_{a_2}^2}{n_{a_2}} - \frac{T_A^2}{n_A} \\
&= \frac{n_{a_2}T_{a_1}^2 + n_{a_1}T_{a_2}^2}{n_{a_1}n_{a_2}} - \frac{(T_{a_1} + T_{a_2})^2}{n_{a_1} + n_{a_2}} \\
&= \frac{(n_{a_2}T_{a_1} - n_{a_1}T_{a_2})^2}{n_{a_1}n_{a_2}n_A} \\
&= \frac{(n_AT_{a_1} - n_{a_1}T_A)^2}{n_{a_1}(n_A - n_{a_1})n_A}.
\end{aligned}$$

Similarly, we have

$$\Delta B = \frac{T_{b_1}^2}{n_{b_1}} + \frac{T_{b_2}^2}{n_{b_2}} - \frac{T_B^2}{n_B} = \frac{(n_BT_{b_1} - n_{b_1}T_B)^2}{n_{b_1}(n_B - n_{b_1})n_B}.$$

Thus, $SS_B(X_iX_j, Y)$ can be updated using $SS_B(X_i, Y)$:

$$SS_B(X_iX_j, Y) = SS_B(X_i, Y) + \Delta A + \Delta B. \tag{2.3}$$

Note that if any one of $\{n_{a_1}, n_{a_2}, n_A\}$ is 0, then $\Delta A = 0$. Similarly, if any one of $\{n_{b_1}, n_{b_2}, n_B\}$ is 0, then $\Delta B = 0$.

Next, we develop an upper bound of $SS_B(X_iX_j, Y)$. We first show the derivation of an upper bound of Δ_A. A similar idea can be applied to find an upper bound of Δ_B.

2.4.2 Bounds on ΔA and ΔB

Let $\{y_m | y_m \in A\} = \{y_{A_1}, y_{A_2}, \cdots, y_{A_{n_A}}\}$ be the phenotype values in group A. Without loss of generality, assume that these phenotype values are arranged in ascending order, i.e.,

$$y_{A_1} \leq y_{A_2} \leq \cdots \leq y_{A_{n_A}}.$$

The derivative of ΔA with respect to T_{a_1} is:

$$\frac{d\Delta A}{dT_{a_1}} = \frac{2n_A(n_A T_{a_1} - n_{a_1} T_A)}{n_{a_1}(n_A - n_{a_1})n_A}.$$

Thus we have

$$\Delta A \text{ monotonically} \begin{cases} \text{increases} & \text{if } T_{a_1} \geq \dfrac{n_{a_1} T_A}{n_A}; \\ \text{decreases} & \text{if } T_{a_1} \leq \dfrac{n_{a_1} T_A}{n_A}. \end{cases}$$

We have the range of T_{a_1}:

$$T_{a_1} \in [l_{a_1}, u_{a_1}] = [\sum_{i=1}^{n_{a_1}} y_{A_i}, \sum_{i=n_A - n_{a_1}+1}^{n_A} y_{A_i}].$$

The maximum value of ΔA is attained when $T_{a_1} = l_{a_1}$ or $T_{a_1} = u_{a_1}$, i.e.,

$$\Delta A \leq \frac{\max\{(n_A l_{a_1} - n_{a_1} T_A)^2, (n_A u_{a_1} - n_{a_1} T_A)^2\}}{n_{a_1}(n_A - n_{a_1})n_A}. \tag{2.4}$$

We use $R_1(X_i X_j Y)$ to denote this upper bound.

Let $\{y_m | y_m \in B\} = \{y_{B_1}, y_{B_2}, \cdots, y_{B_{n_B}}\}$ be the phenotype values in group B. Without loss of generality, assume that these phenotype values are arranged in ascending order, i.e.,

$$y_{B_1} \leq y_{B_2} \leq \cdots \leq y_{B_{n_B}}.$$

Symbols	Formulas
l_{a_1}	$\sum_{i=1}^{n_{a_1}} y_{A_i}$
u_{a_1}	$\sum_{i=n_A-n_{a_1}+1}^{n_A} y_{A_i}$
$R_1(X_iX_jY)$	$\frac{\max\{(n_A l_{a_1} - n_{a_1}T_A)^2, (n_A u_{a_1} - n_{a_1}T_A)^2\}}{n_{a_1}(n_A - n_{a_1})n_A}$
l_{b_1}	$\sum_{i=1}^{n_{b_1}} y_{B_i}$
u_{b_1}	$\sum_{i=n_B-n_{b_1}+1}^{n_B} y_{B_i}$
$R_2(X_iX_jY)$	$\frac{\max\{(n_B l_{b_1} - n_{b_1}T_B)^2, (n_B u_{b_1} - n_{b_1}T_B)^2\}}{n_{b_1}(n_B - n_{b_1})n_B}$

Table 2.2: Notations used in the bounds on ΔA and ΔB

Similarly, we can derive the bound on ΔB:

$$\Delta B \leq \frac{\max\{(n_B l_{b_1} - n_{b_1}T_B)^2, (n_B u_{b_1} - n_{b_1}T_B)^2\}}{n_{b_1}(n_B - n_{b_1})n_B}. \tag{2.5}$$

We use $R_2(X_iX_jY)$ to denote this upper bound. The symbols used in Inequalities (2.4) and (2.5) are summarized in Table 2.2.

From Equation (2.3), Inequalities (2.4) and (2.5), we have the overall upper bound on $SS_B(X_iX_j, Y)$:

Theorem 2.4.1. *(Upper bound of $SS_B(X_iX_j, Y)$)*

$$SS_B(X_iX_j, Y) \leq SS_B(X_i, Y) + R_1(X_iX_jY) + R_2(X_iX_jY).$$

Property 2.4.2. *The upper bound in Theorem 2.4.1 is tight.*

The tightness of the bound is obvious from the derivation of the upper bound, since there exists some genotype of SNP-pair (X_iX_j) that makes the equality hold. For the same reason, we have the following property.

Property 2.4.3. *The upper bound in Theorem 2.4.1 does not exceeds the total sum of squared*

deviations, i.e.,

$$SS_B(X_i, Y) + R_1(X_iX_jY) + R_2(X_iX_jY) \leq SS_T(X_iX_j, Y).$$

2.5 The FastANOVA Algorithm

In this section, we show how our algorithm FastANOVA utilizes the upper bound in Theorem 2.4.1 to achieve efficient two-locus ANOVA testing. In Section 2.5.1, we describe the method for Problem (2) discussed in Section 2.3; that is, given a threshold F_α, we want to find all SNP-pairs whose F-statistics are greater than F_α. Then in Section 2.5.2, we discuss how FastANOVA performs in permutation procedure, i.e., the scenario of Problem (1) in Section 2.3.

2.5.1 A Single Phenotype

Given the threshold F_α, to find all SNP-pairs whose F-statistics are greater than F_α, a brute-force approach is to enumerate all SNP-pairs. To expedite this process, we employ the inequality in Theorem 2.4.1 to prune SNP pairs that will have no chance to pass the significance threshold F_α. From Equation (2.2), we know that finding SNP-pairs (X_iX_j) whose F-statistics $F(X_iX_j, Y) \geq F_\alpha$ is equivalent to finding SNP-pairs satisfying

$$SS_B(X_iX_j, Y) \geq \frac{SS_T(X_i, Y)}{\frac{M-g}{(g-1)F_\alpha} + 1} = \theta.$$

Theorem 2.4.1 suggests that we only need to compute the F-statistics for the SNP-pairs that satisfy:

$$SS_B(X_i, Y) + R_1(X_iX_jY) + R_2(X_iX_jY) \geq \theta.$$

We refer to these SNP-pairs as *candidate* SNP-pairs.

We now discuss how to apply the upper bound in Theorem 2.4.1 in detail. The set of

all SNP-pairs is partitioned into non-overlapping groups such that each group has a common upper bound. For every X_i $(1 \leq i \leq N)$, let $AP(X_i)$ be the set of SNP-pairs

$$AP(X_i) = \{(X_iX_j)|i+1 \leq j \leq N\}.$$

For all SNP-pairs in $AP(X_i)$, n_A, T_A, n_B, T_B and $SS_B(X_i, Y)$ are constants. Moreover, l_{a_1}, u_{a_1} are determined by n_{a_1}, and l_{b_1}, u_{b_1} are determined by n_{b_1}. Therefore, in the upper bound, n_{a_1} and n_{b_1} are the only variables that depend on X_j and may vary for different SNP-pairs (X_iX_j) in $AP(X_i)$.

Note that n_{a_1} is the number of 1's in X_j when X_i takes value 1, and n_{b_1} is the number of 1's in X_j when X_i takes value 0. It is easy to prove that switching n_{a_1} and n_{a_2} does not change the F-statistic value and the correctness of the upper bound. This is also true if we switch n_{b_1} and n_{b_2}. Therefore, without loss of generality, we can always assume that n_{a_1} is the smaller one between the number of 1's and number of 0's in X_j when X_i takes value 1, and n_{b_1} is the smaller one between the number of 1's and number of 0's in X_j when X_i takes value 0.

For example, using the dataset showing in Table 1.1, for SNP-pair (X_iX_2), $n_a = 1$ since the minimum of number of 1's and 0's in X_2 when $X_1 = 1$ is 1 (the number of 1's), and $n_b = 2$ since the minimum of number of 1's and 0's in X_2 when $X_1 = 0$ is 2 (the number of 0's).

The following property specifies the values that n_{a_1} and n_{b_1} can take. The proof is straightforward and omitted here.

Property 2.5.1. *If there are m 1's and $(M-m)$ 0's in X_i, then for any $(X_iX_j) \in AP(X_i)$, the possible values that n_{a_1} can take are $\{0, 1, 2, \cdots, \lfloor m/2 \rfloor\}$. The possible values that n_{b_1} can take are $\{0, 1, 2, \cdots, \lfloor (M-m)/2 \rfloor\}$.*

To efficiently retrieve the candidates, the SNP-pairs (X_iX_j) in $AP(X_i)$ are grouped by their (n_{a_1}, n_{b_1}) values and indexed in a 2D array, referred to as $Array(X_i)$.

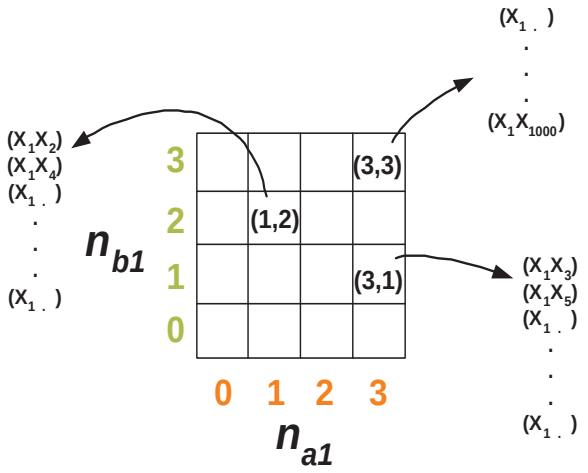


Figure 2.2: The index array $Array(X_1)$ for efficient retrieval of the candidate SNP-pairs

Example 2.5.2. *Using the example dataset shown in Table 1.1, we consider the SNP-pairs in* $AP(X_1)$*, i.e.,* $\{(X_1X_2), (X_1X_3), (X_1X_4), (X_1X_5), \cdots, (X_1X_{1000})\}$*. There are 12 individuals in the dataset.* X_1 *contains 6 0's and 6 1's. Therefore, the possible values of* n_{a_1} *and* n_{b_1} *are* $\{0, 1, 2, 3\}$*. Figure 3.2 shows the* 4×4 *array,* $Array(X_1)$*, whose entries represent the possible values of* (n_{a_1}, n_{b_1}) *for the SNP-pairs in* $AP(X_i)$*. The entries in the same column have the same* n_{a_1} *value. The entries in the same row have the same* n_{b_1} *value. The* n_{a_1} *value of each column is noted beneath each column. The* n_{b_1} *value of each row is noted left to each row. Each entry of the array is a pointer to the SNP-pairs having the corresponding* (n_{a_1}, n_{b_1}) *values. For example, for SNP-pair* (X_1X_3)*, its* $(n_{a_1}, n_{b_1}) = (3, 1)$*. Thus it is indexed by entry (3,1).*

Note that for a SNP-pair $(X_iX_j) \in AP(X_i)$, n_{a_1} and n_{a_2} can be calculated faster than performing the two-locus ANOVA test. To obtain n_{a_1} and n_{a_2}, we only need to count the numbers of 0's and 1's of X_j when X_i is equal to 0 and 1 respectively, which can be done by a linear scan of the $M \times 2$ binary matrix consisting of the genotypes of X_i and X_j. In contrast, to calculate the F-statistic, we first need to scan the $M \times 3$ binary matrix consisting of X_i, X_j and Y in order to find out how the phenotype values are grouped by the genotype of (X_iX_j). Then a constant time $O(t)$ is required to compute the F-statistic.

Property 2.5.3. *For any SNP X_i, the maximum number of the entries in $Array(X_i)$ is*

$$(\lceil \frac{M}{4} \rceil + 1)^2.$$

The proof of Property 2.5.3 is straightforward and omitted here. In order to find candidate SNP-pairs, we scan all entries in $Array(X_i)$ to calculate their upper bounds. Since the SNP-pairs indexed by the same entry share the same (n_{a_1}, n_{b_1}) value, they have the same upper bound.

Property 2.5.4. *Given phenotype Y, for any SNP X_i, the SNP-pairs indexed by the same entry in $AP(X_i)$ have the same upper bound value.*

For typical genome-wide association studies, the number of individuals M is much smaller than the number of SNPs N. From Property 2.5.3, there must be a group of SNP-pairs indexed by the same entry of $AP(X_i)$. In Example 2.5.2, there are in total 16 entries in $Array(X_1)$, and 999 SNP-pairs in $AP(X_1)$. Thus many SNP-pairs share the same (n_{a_1}, n_{b_1}) value and hence indexed by the same entry in $Array(X_1)$. Moreover, from Property 2.5.4, we can calculate the upper bound for the group of SNP-pairs indexed by the same entry together. It is these two key properties of the index structure that help to reduce the complexity of the algorithm. The additional cost for accessing $Array(X_i)$ is minimal compared to performing ANOVA tests for all pairs $(X_iX_j) \in AP(X_i)$ since $M \ll N$.

Algorithm 1 describes the FastANOVA algorithm for finding the SNP-pairs whose F-statistics are greater than the threshold F_α. The inputs of FastANOVA include the N SNPs, the phenotype Y and the critical value F_α. For each X_i, FastANOVA first indexes $(X_iX_j) \in AP(X_i)$ using $Array(X_i)$. Then it retrieves the candidate SNP-pairs by accessing $Array(X_i)$ and records them in $Cand(X_i, Y)$. The candidates in $Cand(X_i, Y)$ are then evaluated for their F-statistics. The candidates whose F-statistics are greater than or equal to F_α are reported by the algorithm.

Algorithm 1: FastANOVA (no phenotype permutation)

Input: SNPs $X' = \{X_1, X_2, \cdots, X_N\}$, phenotype Y, and threshold F_α

Output: find the set of SNP-pairs

$$Result(Y) = \{(X_iX_j) | F(X_iX_j, Y) \geq F_\alpha, 1 \leq i < j \leq N\}$$

```
for every X_i ∈ X', do
    index (X_iX_j) ∈ AP(X_i) by Array(X_i);
    access Array(X_i) to find the candidate SNP-pairs and store them in Cand(X_i, Y);
    for every (X_iX_j) ∈ Cand(X_i, Y) do
        if F(X_iX_j, Y) ≥ F_α then
            Result(Y) ← (X_iX_j);
        end
    end
end
return Result(Y).
```

2.5.2 Permutation Procedure

For multiple tests, permutation procedure is often used in genetic analysis for controlling family-wise error rate. For genome-wide association study, permutation is less commonly used because it often entails prohibitively long computation time. Our FastANOVA algorithm makes permutation procedure feasible in genome-wide association study.

Let $Y' = \{Y_1, Y_2, \cdots, Y_K\}$ be K permutations of the phenotype Y. Following the idea discussed in Section 2.5.1, the upper bound in Theorem 2.4.1 can be easily incorporated in the algorithm to handle the permutations.

Property 2.5.5. *For every SNP X_i, the indexing structure $Array(X_i)$ is independent of the permuted phenotypes in Y'.*

The correctness of this property relies on the fact that, for any $(X_iX_j) \in AP(X_i)$, n_{a_1} and n_{b_1} only depend on the genotype of the SNP-pair and thus remain constant for different phenotype permutations. Therefore, for each X_i, once we build $Array(X_i)$, it can be reused in all permutations.

The FastANOVA algorithm for permutation test is described in Algorithm 2. The inputs include the N SNPs, K phenotype permutations, and the Type I error threshold α. The goal

Algorithm 2: FastANOVA (for permutation test)

Input: SNPs $X' = \{X_1, X_2, \cdots, X_N\}$, phenotype permutations $Y' = \{Y_1, Y_2, \cdots, Y_K\}$, and the Type I error α

Output: find the critical value F_α

```
Tlist ← αK dummy phenotype permutations with F-statistics 0 ;
F_α = 0;
for every X_i ∈ X', do
    index (X_iX_j) ∈ AP(X_i) by Array(X_i);
    for every Y_k ∈ Y', do
        access Array(X_i) to find the candidate SNP-pairs and store them in
        Cand(X_i, Y_k);
        for every (X_iX_j) ∈ Cand(X_i, Y_k) do
            if F(X_iX_j, Y_k) ≥ F_α then
                update Tlist;
                F_α = the smallest test value in Tlist;
            end
        end
    end
end
return F_α.
```

is to find the critical value F_α, which is the αK-th largest value in $\{F_{Y_k} | Y_k \in Y'\}$. Recall that F_{Y_k} is the maximum F-statistic value for phenotype Y_k. We use $Tlist$ to keep the αK phenotype permutations having the largest F-statistics found by the algorithm so far. Initially, $Tlist$ contains αK dummy phenotype permutations with test values 0. The smallest F-statistic value in $Tlist$, initially 0, is used as the threshold to prune the SNP-pairs. For each X_i, FastANOVA first indexes $(X_iX_j) \in AP(X_i)$ using $Array(X_i)$. Then it finds the set of candidate SNP-pairs $Cand(X_i, Y_k)$ by accessing $Array(X_i)$ for every phenotype permutation Y_k. The candidates in $Cand(X_i, Y_k)$ are then evaluated for their F-statistics. If a candidate's F-statistic value is greater than the current threshold, then $Tlist$ is updated accordingly: If the candidate's phenotype Y_k is not in the $Tlist$, then the phenotype in $Tlist$ having the smallest F-statistic value is replaced by Y_k. If the candidate's phenotype Y_k is already in $Tlist$, we only need to update its corresponding F-statistic value to be the maximum value found for the phenotype so far. The threshold is also updated to be the smallest F-statistic value in $Tlist$.

2.5.3 Complexity Analysis

In this section, we study the time and space complexities of the FastANOVA algorithm for permutation test. The complexity for a single phenotype can be analyzed in a similar way.

Time complexity: For each X_i, FastANOVA needs to index (X_iX_j) in $AP(X_i)$. The complexity to build the indexing structure for all SNPs is $O(N(N-1)M/2)$. The worst case for accessing all $Array(X_i)$ for all permutations is $O(N \times K \times (\lceil \frac{M}{4} \rceil + 1)^2) = O(NKM^2)$. Let $C = \sum_{i,k} |Cand(X_i, Y_k)|$ represent the total number of candidates. The overall time complexity of FastANOVA is thus $O(N(N-1)M/2) + O(NK \times (\lceil \frac{M}{4} \rceil + 1)^2) + O(\sum_{i,k} |Cand(X_i, Y_k)|M) = O(N^2M + NKM^2 + CM)$. The experimental results show that the overhead of building the indexing structures and accessing them for candidate retrieval are negligible when large permutation tests are needed. The time complexity of the brute-force approach is $O(KN(N-1)M/2) = O(KN^2M)$. Note that in a typical genotype-phenotype association study, the number of SNPs N is much lager than the number of individuals M. Therefore, when the number of permutations K is large, e.g. thousands, the complexity of FastANOVA is much less than the complexity of the brute force approach.

Space complexity: The total number of variables in the dataset, including the SNPs and the phenotype permutations, is $N + K$. The maximum space of the indexing structure $Array(X_i)$ is $O((\lceil \frac{M}{4} \rceil + 1)^2 + N)$. Note that for each SNP X_i, FastANOVA only needs to access one indexing structure, $Array(X_i)$, for all permutations. Once the evaluation process for X_i is done for all permutations, $Array(X_i)$ can be cleared from the memory. Therefore, the space complexity of FastANOVA is $O((N+K)M) + O((\lceil \frac{M}{4} \rceil + 1)^2 + N) = O((N+K)M)$ since $M \ll N$. The space complexity is linear to the dataset size.

2.6 Experimental Results

In this section, we present extensive experimental results on evaluating the performance of the FastANOVA algorithm. We show (1) the runtime comparison between FastANOVA and

	cardiovascular	metabolism	neurosensory
# individuals	19	26	34
# SNPs	14,513	43,856	66,006

Table 2.3: Statistics of the SNP datasets

the brute-force approach under various experimental settings, (2) the punning effect of the upper bound, and (3) the relative computational cost of each component of FastANOVA. FastANOVA is implemented in C++. The experiments are performed on a 2.4 GHz PC with 1G memory running WindowsXP system.

Dataset: The SNP dataset used for the experiments is extracted from a set of combined SNPs from the 140k Broad/MIT mouse dataset ($http://www.broad.mit.edu/$) and 10k GNF mouse dataset ($http://www.gnf.org/$). This merged dataset has 156,525 SNPs for 71 individuals. The missing values in the dataset are imputed using NPUTE (Roberts et al. (2007)). We use both real phenotypes and synthetic phenotypes in our experiments. The real phenotype data is available from the Jackson Laboratory ($http://www.jax.org/$).

2.6.1 Real Phenotypes

We use three real phenotypes in our experiments: cardiovascular (blood pressure), metabolism (water intake), and neurosensory (acoustic startle response). Table 2.3 shows the statistics of the genotype datasets corresponding to the three phenotypes. The number of SNPs in the table indicates the number of unique SNPs in each genotype dataset.

We first show the results on finding the critical value F_α, which is more time-consuming than finding the significance SNP-pairs given the critical value F_α for a single phenotype.

Finding critical value F_α

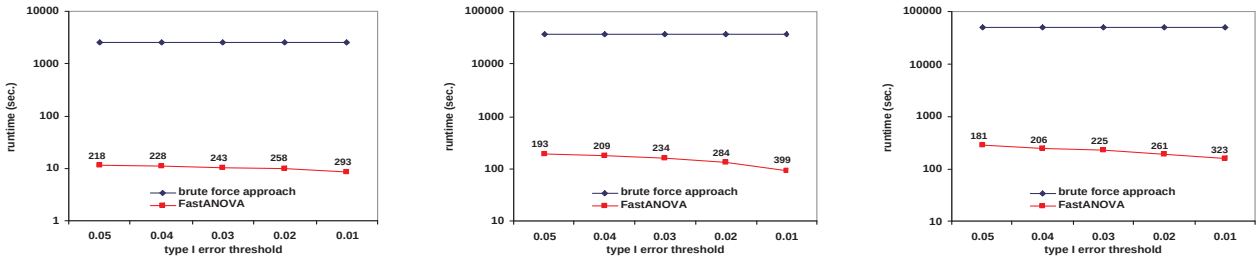
FastANOVA v.s. the brute-force approach We compare FastANOVA with the brute-force approach under various experimental settings. Since the brute-force approach is very time-consuming, we use a moderate number of SNPs and permutations in the default setting in

order to show the performance comparisons. The default setting is as follows: The Type I error threshold $\alpha = 0.01$. The number of permutations is 100. The number of SNP is 10,000 for the two larger datasets of metabolism and neurosensory, and 2,900 for the cardiovascular SNP dataset. These experimental settings are chosen to demonstrate the performance gain and enhanced scalability offered by FastANOVA over the brute-force implementation. FastANOVA can handle much larger SNP panels and larger number of permutation tests. The performance of FastANOVA is expected to follow the same trends presented in the remainder of this section.

Figures 2.3, 2.4, and 2.5 show the running time comparison of FastANOVA and the brute-force approach on the three genotype phenotype datasets using different settings. The y-axis is in logarithm scale. The numbers above the runtime line of FastANOVA indicate the ratio of the runtimes of the brute-force approach over FastANOVA. We terminate the programs that have run over 72 hours without completion.

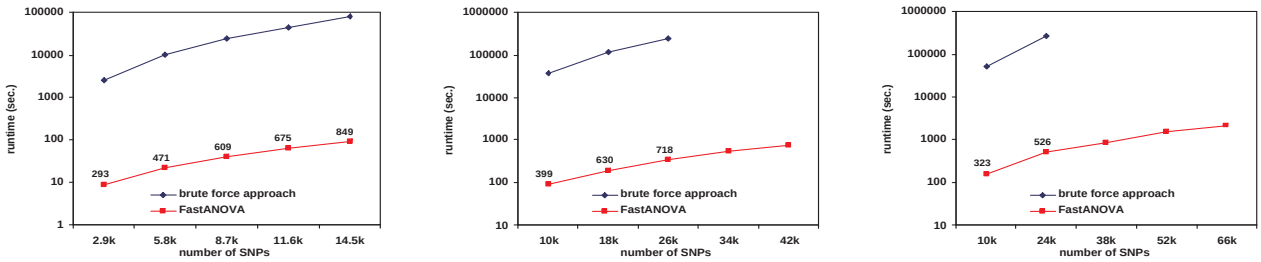
Figure 2.3 shows the runtime comparison when varying the Type I error thresholds. For each dataset, the runtime of the brute-force approach does not change over different Type I error thresholds. The runtime of FastANOVA decreases as the threshold decreases. FastANOVA offers 218 fold speedup when $\alpha = 0.05$ and 293 fold speedup when $\alpha = 0.01$ on cardiovascular dataset. We can also observe a similar two-orders-of-magnitude speedup in the metabolism and neurosensory datasets. This is consistent with the pruning effect of the upper bound, which will be presented later in this section. In general, the lower the Type I error threshold, the more powerful the pruning effect, hence the faster the algorithm.

Figure 2.4 depicts the comparison of these two approaches when the number of SNPs changes. From these figures, it is clear that FastANOVA is about two orders of magnitude faster than the brute-force approach. The brute-force approach cannot finish in 72 hours when the number of unique SNPs is greater than 26k in the metabolism dataset and greater than 24k in the neurosensory dataset. We observe that the runtime ratio tends to increase (approaching three-orders-of-magnitude speedup) as the number of SNPs increases. This indicates that the



(a) cardiovascular (b) metabolism (c) neurosensory

Figure 2.3: Performance comparison between FastANOVA and the brute-force approach when varying Type I error thresholds



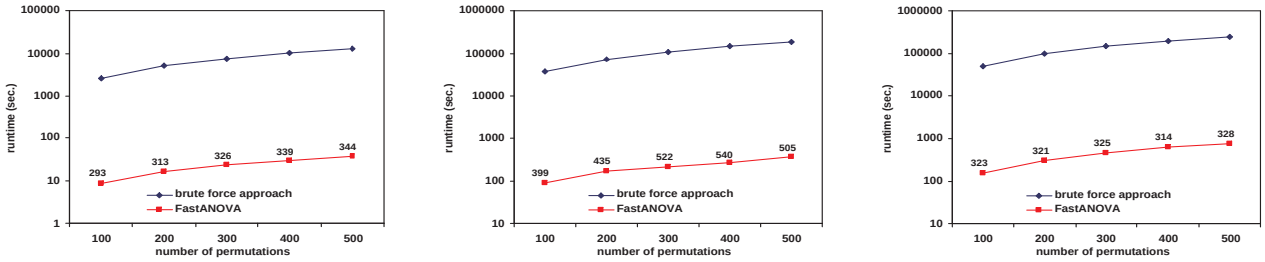
(a) cardiovascular (b) metabolism (c) neurosensory

Figure 2.4: Performance comparison between FastANOVA and the brute-force approach when varying the number of SNPs

performance gain of FastANOVA is even higher for larger SNP datasets.

Figure 2.5 shows the runtime comparison when the number of phenotype permutations changes. The runtime of the brute-force approach is linear with respect to the number of permutations. FastANOVA is consistently two orders of magnitude faster than the brute-force approach. The performance gap increases as the number of permutations increases.

Pruning effect of the upper bound Table 2.4 shows the percentage of SNP-pairs pruned under different experimental settings. Since the three datasets have different numbers of SNPs, the 1st to 5th rows in the category of "# SNPs" correspond to the settings from left to right on x-axis in each plot in Figure 2.4. Most SNP-pairs are pruned under all settings. Moreover, as the Type I error threshold α decreases, the pruning ratio increases, which is consistent with runtime comparison shown in Figure 2.3. As the number of SNPs increases, the pruning ratio



(a) cardiovascular (b) metabolism (c) neurosensory

Figure 2.5: Performance comparison between FastANOVA and the brute-force approach when varying the number of permutations

also increases. This is because, with more SNPs, the dynamic threshold used to prune the search space becomes higher. Hence a larger portion of SNPs are pruned. This is consistent with results shown in Figure 2.4. Note that from Table 2.4 we observe that the pruning ratio tends to remain steady when the number of permutations changes. However, we observe that the runtime ratio increases as the number of permutations increases. The reason for these two different trends will become clear after we show the results on the computational cost of each component of FastANOVA in the next subsection.

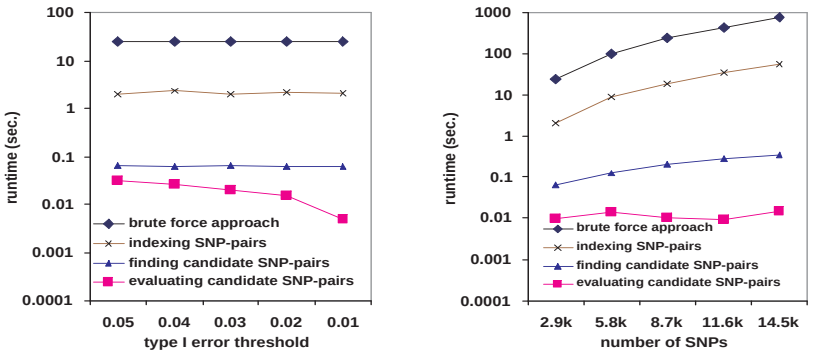
Finding significant SNP-pairs

In this subsection, we study the comparison between FastANOVA and the brute-force approach in finding significant SNP-pairs given a critical value F_α. Only the original phenotype (without permutations) is used in this procedure. We examine the detailed computation cost of each component of the FastANOVA algorithm. FastANOVA has three major components: building the indexing structure $Array(X_i)$ for every SNP X_i, accessing $Array(X_i)$ to find the candidate SNP-pairs, and performing ANOVA tests on these candidates.

Figures 2.6 to 2.8 show the performance comparison on the three datasets. The default experimental setting is the same as before. We examine the performance on metabolism dataset in detail. Similar behaviors can be observed on the other two datasets. Figure 2.7(a) and Figure 2.7(b) show the runtime of these three components when varying the Type I error threshold

		cardiovascular	metabolism	neurosensory
α	0.05	99.881%	99.724%	99.701%
	0.04	99.907%	99.758%	99.751%
	0.03	99.928%	99.797%	99.792%
	0.02	99.949%	99.877%	99.853%
	0.01	99.974%	99.929%	99.911%
# SNPs	1st	99.974%	99.929%	99.911%
	2nd	99.991%	99.985%	99.979%
	3rd	99.996%	99.996%	99.997%
	4th	99.998%	99.996%	99.997%
	5th	99.998%	99.993%	99.998%
# Perm.	100	99.974%	99.929%	99.911%
	200	99.966%	99.935%	99.917%
	300	99.977%	99.962%	99.919%
	400	99.977%	99.961%	99.914%
	500	99.974%	99.953%	99.907%

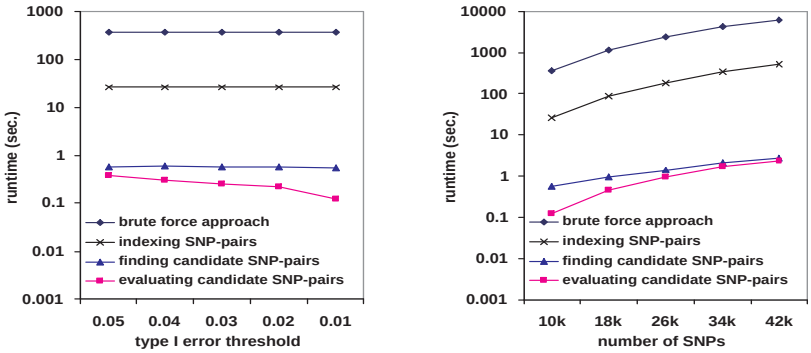
Table 2.4: Pruning effects on cardiovascular, metabolism and neurosensory datasets when finding critical value F_α



(a) Varying threshold values (b) Varying number of SNPs

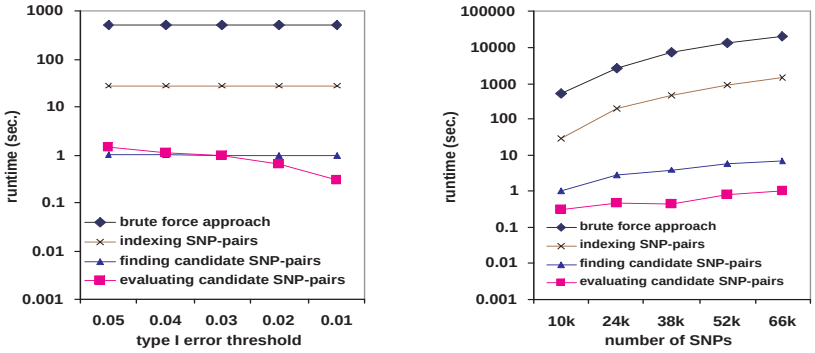
Figure 2.6: Finding significant SNP-pairs (cardiovascular dataset)

and number of SNPs in the metabolism dataset respectively. Since F_α is a function of α, in Figure 2.7(a), we plot the runtime with respect to α. In both figures, the three lines from the bottom show the runtime of these three components. The runtime of the brute-force approach is the top line. As we can see from these two figures, performing two-locus ANOVA tests on candidate SNP pairs is two to three orders of magnitude faster than performing such tests on all SNP-pairs. This is the benefit of the upper bound pruning since most SNP-pairs have been



(a) Varying threshold values (b) Varying number of SNPs

Figure 2.7: Finding significant SNP-pairs (metabolism dataset)



(a) Varying threshold values (b) Varying number of SNPs

Figure 2.8: Finding significant SNP-pairs (neurosensory dataset)

pruned and only a very small portion of candidates need to be evaluated for their F-statistics. The cost for accessing the indexing structures is also small, which demonstrates the efficiency of the method introduced in Section 3.4.1 for candidate retrieval. Among the three components of FastANOVA, the most time-consuming one is building the index structures. Yet, its runtime is only a small fraction of the runtime of performing the two-locus ANOVA tests on all SNP pairs. Note that, in permutation test, building the index structures is a one time cost. Once the index structures are built, they can be reused in all permutations. Therefore, the amortized overhead per permutation decreases when the number of permutations increases. This is why the pruning ratio remains steady in Table 2.4 while the runtime ratio increases in Figure 2.5 when the number of permutations increases.

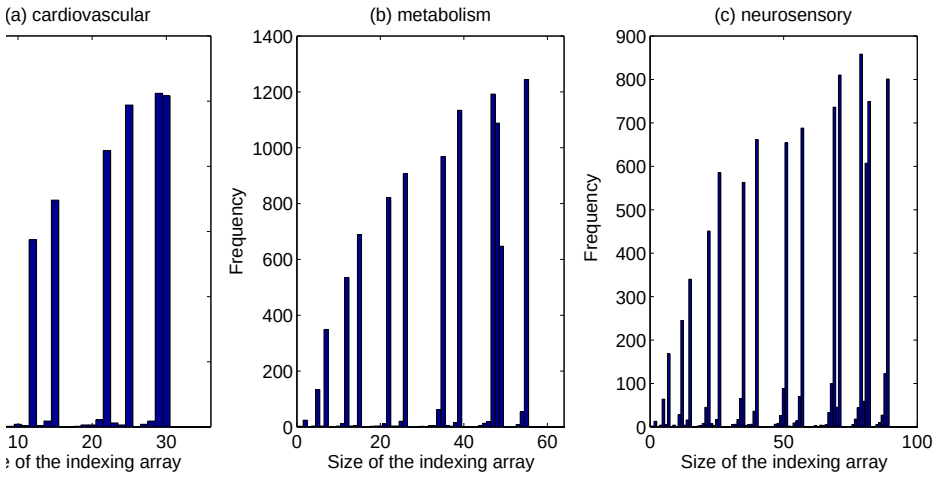


Figure 2.9: Histogram of the sizes of the indexing structures

cardiovascular	metabolism	neurosensory
97.865%	97.844%	98.061%

Table 2.5: Pruning effect on cardiovascular, metabolism and neurosensory datasets when finding F_{Y_k} for all permutations

Figure 2.9 shows the histogram of the sizes of the indexing structures for the three datasets. From Property 2.5.3, the maximum sizes of the indexing structures are 36 for the cardiovascular dataset, 64 for the metabolism dataset, and 100 for the neurosensory dataset. It is clear from the figure that the actual sizes of the indexing structures are much smaller than the maximum sizes.

Finding F_{Y_k} for all permutations

Sometimes users may be interested in finding F_{Y_k} values of all phenotype permutations. In this way, the users can get the critical value F_α for any Type I error threshold α ranging from 0 to 1, without re-running the permutation tests for different thresholds. Recall that, given a set of phenotype permutations $Y' = \{Y_1, Y_2, \cdots, Y_K\}$, $F_{Y_k} = \max\{F(X_i X_j, Y_k) | 1 \leq i < j \leq N\}$ is the maximum F-statistic value for permutation Y_k. F_α is the αK-th largest value in $\{F_{Y_k} | Y_k \in Y'\}$. In this subsection, we show the pruning effect of the upper bound when it is applied to determine F_{Y_k} for every Y_k ($1 \leq k \leq K$). Note that in this case, for each

		uniform	normal	exponential
α	0.05	96.469%	97.793%	99.335%
	0.04	96.888%	98.222%	99.401%
	0.03	97.695%	98.631%	99.502%
	0.02	98.712%	99.072%	99.617%
	0.01	99.605%	99.506%	99.737%
# SNPs	10k	99.605%	99.506%	99.737%
	22k	99.864%	99.814%	99.924%
	34k	99.907%	99.905%	99.967%
	46k	99.928%	99.889%	99.965%
	58k	99.941%	99.942%	99.963%
# Perm.	100	99.605%	99.506%	99.737%
	200	98.891%	99.398%	99.726%
	300	98.897%	99.072%	99.780%
	400	98.623%	99.315%	99.762%
	500	98.709%	99.199%	99.759%
# indiv.	28	99.756%	99.695%	99.893%
	30	99.422%	99.577%	99.880%
	32	99.605%	99.506%	99.737%
	34	99.073%	99.289%	99.773%
	36	98.736%	98.832%	99.745%

Table 2.6: Pruning effect when finding critical value F_α using three synthetic phenotypes

permutation Y_k, the dynamic threshold used to prune the search space is the largest F-statistic value of Y_k identified by the algorithm so far.

Table 2.5 shows the pruning ratio of applying the upper bound to the three real phenotype datasets. The experimental setting is the same as the default setting before. As expected, the pruning ratios are slightly lower than those in Table 2.4, where smaller Type I error thresholds are used to prune the search space. However, the pruning ratios on all three datasets are still above 97%. Moreover, finding all F_{Y_k} provides the advantage that we can get the F_α values for all possible α values instead of just for a specific one.

2.6.2 Synthetic Phenotypes

To further study the performance of FastANOVA, we generate three synthetic phenotypes whose values follow three different distributions: uniform, standard normal, and standard

exponential distribution. Our purpose is to study the pruning effect of the upper bound under different phenotype distributions. The default setting of the experiments in this subsection is as follows: #individuals = 32, #SNPs=10,000, #permutations=100, $\alpha = 0.01$. There are 60,970 unique SNPs for these 32 individuals.

Table 2.6 shows the pruning ratio of FastANOVA under different settings using permutation test. In this table, we also include the pruning ratio when the number of individuals varies. We observe that the pruning effects are similar to that of real phenotypes, which indicates that the upper bound pruning is effective and insensitive to different phenotype distributions.

2.7 Conclusion

The large number of available SNPs poses great computational challenge to the genome-wide association study. To assess the significance of the findings, permutation test is usually required. These factors make the association study a very time-consuming process. Thus tools that can improve the efficiency of the association study are in demand.

In this chapter we present an efficient algorithm, FastANOVA, for genome-wide two-locus ANOVA test. FastANOVA is an exhaustive algorithm which guarantees to find the optimal solution. Experimental results demonstrate that FastANOVA is two to three orders of magnitude faster than the brute-force alternative. The efficiency of FastANOVA is gained from two sources. First, it utilizes an upper bound of the two-locus ANOVA test value to prune a majority of the SNP-pairs. Second, it identifies and reuses computation units that are independent of the phenotype and hence are invariant in permutation test. By eliminating redundant computation of these invariant units, FastANOVA is much more efficient than the brute-force method.

Chapter 3

The FastChi Algorithm

3.1 Introduction

As our initial attempt to develop scalable algorithms for genome-wide association study, FastANOVA is specifically designed for the ANOVA test on quantitative phenotypes. Another category of phenotypes is generated in case-control study, where the phenotypes are binary variables representing disease/non-disease individuals. Chi-square test is one of the most commonly used statistics in binary phenotype association study. We can extend the principles in FastANOVA for efficient two-locus chi-square test (Zhang et al. (2009)). The general idea of FastChi is similar to that of FastANOVA, i.e., re-formulating the chi-square test statistic to establish an upper bound of two-locus chi-square test, and indexing the SNP-pairs according to their genotypes in order to effectively prune the search space and reuse redundant computations. In this chapter, we introduce the FastChi algorithm.

FastChi is an *exhaustive* algorithm, i.e., it guarantees to find the optimal solution, though it does not explicitly examine all possible SNP-pairs. A large portion of the SNP-pairs are pruned without the need of performing the tests. FastChi establishes an upper bound of the two-locus chi-square test. The upper bound is the sum of two terms: one based on the single-locus chi-square test, and the other based on the pair-wise SNP genotypes. The computational cost of deriving this bound is much less than the cost of performing a two-locus chi-square

(a) Contingency table for $\chi^2(X_i, Y)$

	$X_i = 0$	$X_i = 1$	Total
$Y = 0$	event A	event B	
$Y = 1$	event C	event D	
Total			M

(b) Contingency table for $\chi^2(X_iX_j, Y)$

	$X_i = 0$		$X_i = 1$		Total
	$X_j = 0$	$X_j = 1$	$X_j = 0$	$X_j = 1$	
$Y = 0$	event a_1	event a_2	event b_1	event b_2	
$Y = 1$	event c_1	event c_2	event d_1	event d_2	
Total					M

Table 3.1: Contingency tables for chi-square testing

test. Consequently, FastChi is able to identify SNP pairs with significant chi-square values for a given phenotype using only a small fraction of the time required by performing two-locus chi-square test on all SNP pairs.

3.2 The Problem

Let $\{X_1, X_2, \cdots, X_N\}$ be the set of all biallelic SNPs , and Y be the binary phenotype of interest (e.g., disease or non-disease). We adopt the convention of using 0 to represent majority allele and 1 to represent minority allele, and use 0 for non-disease and 1 for disease. For any SNP X_i ($1 \leq i \leq N$), we represent the chi-square test value of X_i and Y as $\chi^2(X_i, Y)$. For any SNP-pair X_i and X_j, we use $\chi^2(X_iX_j, Y)$ to represent the chi-square test value for the combined effect of (X_iX_j) with phenotype Y. Table 3.0(a) and 3.0(b) show example contingency tables for calculating $\chi^2(X_i, Y)$ and $\chi^2(X_iX_j, Y)$ when X_i, X_j and Y are binary variables.

We formalize the problem as follows. Given a set of N biallelic SNPs $\{X_1, X_2, \cdots, X_N\}$ and a binary phenotype Y for a set of M individuals. Let $Y' = \{Y_1, Y_2, \cdots, Y_K\}$ be the set of K permutations of Y. There are two possible cases:

(1) For a single pass association study, i.e., no permutation correction needed: find all

SNP-pairs (X_iX_j) such that

$$\chi^2(X_iX_j, Y) \geq \theta.$$

(2) In the case where there are multiple phenotype permutations: for each $Y_k \in Y'$, find all SNP-pairs (X_iX_j) such that

$$\chi^2(X_iX_j, Y_k) \geq \theta, (1 \leq k \leq K).$$

Note that if $Y' = \{Y\}$ then cases (1) and (2) are the same. Case (1) is actually a special case of (2). Our problem formalization can also be applied in other problem settings. For example, it is easy to modify this problem definition as finding the top-k SNP-pairs that have the largest chi-square test values among all SNP-pairs. In this scenario, θ would be a dynamic value, i.e., the k-th largest chi-square test value identified by the algorithm so far.

3.3 The Upper Bound

3.3.1 Updating Chi-square Statistic

In this section, we show that for any two SNPs, X_i and X_j, $\chi^2(X_iX_j, Y)$ can be derived from $\chi^2(X_i, Y)$. The results in this section provide the foundation for developing the upper bound of $\chi^2(X_iX_j, Y)$ which will be presented in Section 3.3.2.

Let $A, B, C, D, a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2$ represent the events as shown in Table 3.0(a) and Table 3.0(b). Let E_{event} and O_{event} denote the expected value and observed value of certain event. $\chi^2(X_i, Y)$ and $\chi^2(X_iX_j, Y)$ can be calculated as follows:

$$\chi^2(X_i, Y) = \sum_{event \in \{A,B,C,D\}} \frac{(O_{event} - E_{event})^2}{E_{event}},$$

$$\chi^2(X_iX_j, Y) = \sum_{event \in \{a_1,a_2,b_1,b_2,c_1,c_2,d_1,d_2\}} \frac{(O_{event} - E_{event})^2}{E_{event}}.$$

For event A, its corresponding component in $\chi^2(X_i, Y)$ calculation is

$$\frac{(O_A - E_A)^2}{E_A}.$$

For events a_1 and a_2, their corresponding component in $\chi^2(X_i X_j, Y)$ calculation is

$$\frac{(O_{a_1} - E_{a_1})^2}{E_{a_1}} + \frac{(O_{a_2} - E_{a_2})^2}{E_{a_2}}.$$

Note that $O_A = O_{a_1} + O_{a_2}$, and $E_A = E_{a_1} + E_{a_2}$. The difference between these two components is

$$\begin{aligned}
\Delta_A &= \frac{(O_{a_1} - E_{a_1})^2}{E_{a_1}} + \frac{(O_{a_2} - E_{a_2})^2}{E_{a_2}} - \frac{(O_A - E_A)^2}{E_A} \\
&= \frac{M(O_{a_1}O_{c_2} - O_{a_2}O_{c_1})^2}{(O_A + O_B)(O_A + O_C)(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})}.
\end{aligned}$$

Similarly, for events C, c_1 and c_2, we have

$$\begin{aligned}
\Delta_C &= \frac{(O_{c_1} - E_{c_1})^2}{E_{c_1}} + \frac{(O_{c_2} - E_{c_2})^2}{E_{c_2}} - \frac{(O_C - E_C)^2}{E_C} \\
&= \frac{M(O_{c_1}O_{a_2} - O_{c_2}O_{a_1})^2}{(O_C + O_D)(O_A + O_C)(O_{c_1} + O_{a_1})(O_{c_2} + O_{a_2})}.
\end{aligned}$$

Adding Δ_A and Δ_C together, we have $\Delta_A + \Delta_C$

$$= \frac{M^2(O_{a_1}O_{c_2} - O_{a_2}O_{c_1})^2}{(O_A + O_B)(O_A + O_C)(O_C + O_D)(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})}.$$

Let

$$T_1 = \frac{M^2}{(O_A + O_B)(O_A + O_C)(O_C + O_D)}.$$

$\Delta_A + \Delta_C$ can be rewritten as

$$\Delta_A + \Delta_C = \frac{T_1(O_{a_1}O_{c_2} - O_{a_2}O_{c_1})^2}{(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})}. \tag{3.1}$$

Similarly, let Δ_B be the difference between the components for events B, b_1 and b_2 in $\chi^2(X_i, Y)$ and $\chi^2(X_iX_j, Y)$ calculations, and Δ_D be the difference between the components for events D, d_1, and d_2 in $\chi^2(X_i, Y)$ and $\chi^2(X_iX_j, Y)$ calculations. Let

$$T_2 = \frac{M^2}{(O_A + O_B)(O_B + O_D)(O_C + O_D)}.$$

We have

$$\Delta_B + \Delta_D = \frac{T_2(O_{b_1}O_{d_2} - O_{b_2}O_{d_1})^2}{(O_{b_1} + O_{d_1})(O_{b_2} + O_{d_2})}. \tag{3.2}$$

It is easy to see that

$$\chi^2(X_iX_j, Y) = \chi^2(X_i, Y) + \Delta_A + \Delta_C + \Delta_B + \Delta_D.$$

Equations (3.1) and (3.2) show that we can update $\chi^2(X_iX_j, Y)$ using $\chi^2(X_i, Y)$ without computing it from scratch.

3.3.2 Bound on $\Delta_A + \Delta_C$

In this section, we develop an upper bound for $\chi^2(X_iX_j, Y)$. In Section 3.4, we will discuss how this upper bound is utilized by the FastChi algorithm for efficient two-locus chi-square tests.

Following the results from Section 3.3.1, we first show the derivation of the upper bound for $\Delta_A + \Delta_C$. A similar idea can be applied to find the upper bound for $\Delta_B + \Delta_D$.

The bound on $\Delta_A + \Delta_C$ is the result of combining two different bounds, whose derivations

are presented in the following subsections.

Bound 1

Let's look at Equation (3.1) in Section 3.3.1. There are two possible scenarios.

Scenario (1): if $O_{a_1}O_{c_2} \geq O_{a_2}O_{c_1}$, then we have

$\Delta_A + \Delta_C$

$$\begin{aligned}
&= \frac{T_1[(O_A + O_C - O_{a_2} - O_{c_1} - O_{c_2})O_{c_2} - O_{a_2}O_{c_1}]^2}{(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})} \\
&= \frac{T_1[(O_A + O_C)O_{c_2} - (O_{c_1} + O_{c_2})(O_{a_2} + O_{c_2})]^2}{(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})} \\
&\leq \frac{T_1[(O_A + O_C)(O_{c_2} + O_{a_2}) - (O_{c_1} + O_{c_2})(O_{a_2} + O_{c_2})]^2}{(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})} \\
&= \frac{T_1[(O_{a_1} + O_{a_2})(O_{a_2} + O_{c_2})]^2}{(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})} \\
&= T_1(O_{a_1} + O_{a_2})^2 \times \frac{(O_{a_2} + O_{c_2})}{(O_{a_1} + O_{c_1})} = T_1 O_A^2 \times \frac{(O_{a_2} + O_{c_2})}{(O_{a_1} + O_{c_1})}.
\end{aligned}$$

Scenario (2): if $O_{a_1}O_{c_2} < O_{a_2}O_{c_1}$, then we have

$\Delta_A + \Delta_C$

$$
\begin{aligned}
&= \frac{T_1[(O_A + O_C - O_{a_1} - O_{c_1} - O_{c_2})O_{c_1} - O_{a_1}O_{c_2}]^2}{(O_{a_2} + O_{c_1})(O_{a_2} + O_{c_2})} \\
&= \frac{T_1[(O_A + O_C)O_{c_1} - (O_{a_1} + O_{c_1})(O_{c_1} + O_{c_2})]^2}{(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})} \\
&\leq \frac{T_1[(O_A + O_C)(O_{c_1} + O_{c_2}) - (O_{a_1} + O_{c_1})(O_{c_1} + O_{c_2})]^2}{(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})} \\
&= \frac{T_1[(O_{c_1} + O_{c_2})(O_{a_2} + O_{c_2})]^2}{(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})} \\
&= T_1(O_{c_1} + O_{c_2})^2 \times \frac{(O_{a_2} + O_{c_2})}{(O_{a_1} + O_{c_1})} = T_1 O_C^2 \times \frac{(O_{a_2} + O_{c_2})}{(O_{a_1} + O_{c_1})}.
\end{aligned}
$$

Note that O_A and O_C are fixed given X_i and Y. Let $S_1 = \max\{O_A^2, O_C^2\}$. Since $\frac{(O_{a_2} + O_{c_2})}{(O_{a_1} + O_{c_1})} = \left[\frac{O_{X_j=1}}{O_{X_j=0}}|X_i = 0\right]$, we have

$$\Delta_A + \Delta_C \leq T_1 S_1 \left[\frac{O_{X_j=1}}{O_{X_j=0}}|X_i = 0\right]. \tag{3.3}$$

Note that $\left[\frac{O_{X_j=1}}{O_{X_j=0}}|X_i = 0\right]$ is simply the ratio between the number of 1's and the number of 0's in X_j when X_i takes value 0. For example, if the genotypes of X_i are 0000011111 and the genotypes of X_j are 1110010000 across 10 individuals, then $\left[\frac{O_{X_j=1}}{O_{X_j=0}}|X_i = 0\right] = \frac{3}{2}$.

Bound 2

Similarly, we can derive another bound for $\Delta_A + \Delta_C$.

Scenario (1): if $O_{a_1}O_{c_2} \geq O_{a_2}O_{c_1}$, then we have

$\Delta_A + \Delta_C$

$$\begin{aligned} &= \frac{T_1[(O_A + O_C)O_{c_2} - (O_{c_1} + O_{c_2})(O_{a_2} + O_{c_2})]^2}{(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})} \\ &\leq \frac{T_1[(O_A + O_C)(O_{c_2} + O_{c_1}) - (O_{c_1} + O_{c_2})(O_{a_2} + O_{c_2})]^2}{(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})} \\ &= T_1 O_C^2 \times \frac{(O_{a_1} + O_{c_1})}{(O_{a_2} + O_{c_2})}. \end{aligned}$$

Scenario (2): if $O_{a_1}O_{c_2} < O_{a_2}O_{c_1}$, then we have

$\Delta_A + \Delta_C$

$$\begin{aligned} &= \frac{T_1[(O_A + O_C)O_{c_1} - (O_{a_1} + O_{c_1})(O_{c_1} + O_{c_2})]^2}{(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})} \\ &\leq \frac{T_1[(O_A + O_C)(O_{c_1} + O_{a_1}) - (O_{a_1} + O_{c_1})(O_{c_1} + O_{c_2})]^2}{(O_{a_1} + O_{c_1})(O_{a_2} + O_{c_2})} \\ &= T_1 O_A^2 \times \frac{(O_{a_1} + O_{c_1})}{(O_{a_2} + O_{c_2})}. \end{aligned}$$

Similar to Bound 1, combining these two scenarios, we have

$$\Delta_A + \Delta_C \leq T_1 S_1 \left[\frac{O_{X_j=0}}{O_{X_j=1}} | X_i = 0 \right] \tag{3.4}$$

Now let $\mathscr{R}_1 = \min\{ \left[\frac{O_{X_j=1}}{O_{X_j=0}} | X_i = 0 \right], \left[\frac{O_{X_j=0}}{O_{X_j=1}} | X_i = 0 \right] \}$. Combining Bounds (3.3) and (3.4), we have

$$\Delta_A + \Delta_C \leq T_1 S_1 \mathscr{R}_1. \tag{3.5}$$

Inequality (3.5) is the final bound for $\Delta_A + \Delta_C$.

Symbols	Formulas
T_1	$\frac{M^2}{(O_A+O_B)(O_A+O_C)(O_C+O_D)}$
S_1	$\max\{O_A^2, O_C^2\}$
$\mathscr{R}_1$	$\min\{\left[\frac{O_{X_j=1}}{O_{X_j=0}}\|X_i=0\right], \left[\frac{O_{X_j=0}}{O_{X_j=1}}\|X_i=0\right]\}$
T_2	$\frac{M^2}{(O_A+O_B)(O_B+O_D)(O_C+O_D)}$
S_2	$\max\{O_B^2, O_D^2\}$
$\mathscr{R}_2$	$\min\{\left[\frac{O_{X_j=1}}{O_{X_j=0}}\|X_i=1\right], \left[\frac{O_{X_j=0}}{O_{X_j=1}}\|X_i=1\right]\}$

Table 3.2: Notations used in the derivation of the upper bound

3.3.3 Bound on $\Delta_B + \Delta_D$

By analogy, we can get the upper bound for $\Delta_B + \Delta_D$. We present the result here and omit the detailed derivation.

Let $S_2 = \max\{O_B^2, O_D^2\}$, and
$\mathscr{R}_2 = \min\{\left[\frac{O_{X_j=1}}{O_{X_j=0}}|X_i=1\right], \left[\frac{O_{X_j=0}}{O_{X_j=1}}|X_i=1\right]\}$. We have

$$\Delta_B + \Delta_D \leq T_2 S_2 \mathscr{R}_2 \tag{3.6}$$

3.3.4 The Overall Bound

From Bounds (3.5) and (3.6), we get the overall bound for $\chi^2(X_iX_j, Y)$ as follows:

$$\chi^2(X_iX_j, Y) \leq \chi^2(X_i, Y) + T_1 S_1 \mathscr{R}_1 + T_2 S_2 \mathscr{R}_2. \tag{3.7}$$

The notations used in the upper bound derivation are summarized in Table 3.2.

3.4 The FastChi Algorithm

In this section, we show how our algorithm FastChi utilizes the upper bound derived in Section 3.3.2 to achieve efficient two-locus chi-square testing. In Section 3.4.1, we describe the method for the original phenotype Y. Then in Section 3.4.2, we discuss how FastChi performs under permutation procedure.

3.4.1 One Phenotype

A brute force approach for two-locus tests is to enumerate all SNP-pairs. To expedite this process, we employ Inequality (3.7) to prune SNP pairs that do not have a chance to generate significant chi-square values. The set of all SNP-pairs is partitioned into non-overlapping groups such that Inequality (3.7) can be readily applied to each group. For every X_i $(1 \leq i \leq N)$, let $AP(X_i)$ be the set of SNP-pairs

$$AP(X_i) = \{(X_i X_j) | i+1 \leq j \leq N\}.$$

Inequality (3.7) gives an upper bound of $\chi^2(X_i X_j, Y)$. If this upper bound is smaller than θ (i.e., $\chi^2(X_i, Y) + T_1 S_1 \mathscr{R}_1 + T_2 S_2 \mathscr{R}_2 < \theta$), there is no need to calculate the exact value of $\chi^2(X_i X_j, Y)$, which will be smaller than θ.

We now discuss this idea in detail. For all SNP-pairs in $AP(X_i)$, the phenotype Y and SNP X_i do not vary, thus O_A, O_B, O_C and O_D are constants for the SNP pairs in $AP(X_i)$. The number of individuals, M, is also a constant. Thus, in Inequality (3.7), $T_1 S_1$ and $T_2 S_2$ are constants. Moreover, $\chi^2(X_i, Y)$ is a constant for a given X_i, and θ is given too. Therefore, $\mathscr{R}_1$ and $\mathscr{R}_2$ are the only variables that depend on X_j in Inequality (3.7) and may vary for different SNP-pairs $(X_i X_j) \in AP(X_i)$. Thus for a given X_i, we can treat equation

$$\chi^2(X_i, Y) + T_1 S_1 \mathscr{R}_1 + T_2 S_2 \mathscr{R}_2 = \theta$$

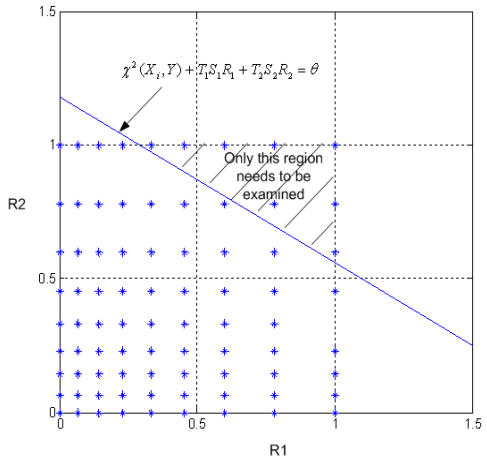


Figure 3.1: Pruning SNP-pairs in $AP(X_i)$ using the upper bound

as a straight line in the 2-D space of $\mathscr{R}_1$ and $\mathscr{R}_2$.

From now on, we use $\mathscr{R}_1(X_iX_j)$ and $\mathscr{R}_2(X_iX_j)$ to represent the specific values of $\mathscr{R}_1$ and $\mathscr{R}_2$ for the SNP-pair (X_iX_j). It is easy to see that $\mathscr{R}_1(X_iX_j)$ and $\mathscr{R}_2(X_iX_j)$ cannot be greater than 1, as summarized in the following property.

Property 3.4.1.

$$\mathscr{R}_1(X_iX_j) = \min\{\left[\frac{O_{X_j=1}}{O_{X_j=0}}|X_i=0\right], \left[\frac{O_{X_j=0}}{O_{X_j=1}}|X_i=0\right]\} \leq 1;$$

$$\mathscr{R}_2(X_iX_j) = \min\{\left[\frac{O_{X_j=1}}{O_{X_j=0}}|X_i=1\right], \left[\frac{O_{X_j=0}}{O_{X_j=1}}|X_i=1\right]\} \leq 1.$$

More specifically, the following property specifies the values that $\mathscr{R}_1(X_iX_j)$ and $\mathscr{R}_2(X_iX_j)$ can take. The proof is straightforward and omitted here.

Property 3.4.2. *If there are m 0's and $(M-m)$ 1's in X_i, then for any $(X_iX_j) \in AP(X_i)$, the possible values that $\mathscr{R}_1(X_iX_j)$ can take are:*

$$\{\frac{0}{m}, \frac{1}{m-1}, \frac{2}{m-2}, \cdots, \frac{\lfloor m/2 \rfloor}{\lceil m/2 \rceil}\}.$$

The possible values that $\mathscr{R}_2(X_iX_j)$ can take are:

$$\{\frac{0}{M-m}, \frac{1}{M-m-1}, \frac{2}{M-m-2}, \cdots, \frac{\lfloor (M-m)/2 \rfloor}{\lceil (M-m)/2 \rceil}\}.$$

Therefore, for all $(X_iX_j) \in AP(X_i)$, in the 2-D space of $\mathscr{R}_1$ and $\mathscr{R}_2$, $(\mathscr{R}_1(X_iX_j), \mathscr{R}_2(X_iX_j))$ fall into the region $[0,1] \times [0,1]$. The line $\chi^2(X_i, Y) + T_1S_1\mathscr{R}_1 + T_2S_2\mathscr{R}_2 = \theta$ divides this region into two parts: one above the line and one below it. Among the SNP-pairs in $AP(X_i)$, we only need to perform the two-locus chi-square test for those ones whose $(\mathscr{R}_1(X_iX_j), \mathscr{R}_2(X_iX_j))$ values are above the line, i.e., whose upper bounds are greater than the threshold θ. We refer to such SNP-pairs as *candidate* SNP-pairs.

Example 3.4.3. *Suppose that there are 32 individuals, half alleles of X_i are 0's, and half are 1's. Thus for the SNP-pairs in $AP(X_i)$, the possible values of $\mathscr{R}_1(X_iX_j)$ (and $\mathscr{R}_2(X_iX_j)$) are $\{\frac{0}{16}, \frac{1}{15}, \frac{2}{14}, \frac{3}{13}, \frac{4}{12}, \frac{5}{11}, \frac{6}{10}, \frac{7}{9}, \frac{8}{8}\}$. Figure 3.1 shows the 2-D space of $\mathscr{R}_1$ and $\mathscr{R}_2$. The blue stars represent the values that $(\mathscr{R}_1(X_iX_j), \mathscr{R}_2(X_iX_j))$ can take. The line $\chi^2(X_i, Y) + T_1S_1\mathscr{R}_1 + T_2S_2\mathscr{R}_2 = \theta$ is also plotted in the figure. The candidate SNP-pairs are those whose $(\mathscr{R}_1(X_iX_j), \mathscr{R}_2(X_iX_j))$ values are in the shaded region. The ones whose $(\mathscr{R}_1(X_iX_j), \mathscr{R}_2(X_iX_j))$ values fall below the line can be pruned without any further test.*

Note that for a SNP-pair $(X_iX_j) \in AP(X_i)$, $\mathscr{R}_1(X_iX_j)$ and $\mathscr{R}_2(X_iX_j)$ can be calculated faster than the two-locus chi-square test. To obtain $\mathscr{R}_1(X_iX_j)$ and $\mathscr{R}_2(X_iX_j)$, we only need to count the numbers of 0's and 1's of X_j when X_i is equal to 0 and 1 respectively, which can be done by a linear scan of the $M \times 2$ binary matrix consisting of the genotypes of X_i and X_j. In contrast, to calculate the two-locus chi-square test value, we first need to scan the $M \times 3$ binary matrix consisting of X_i, X_j and Y in order to fill out the 2×4 contingency table as shown in Table 3.0(b). Then a constant time $O(t)$ is required to compute the chi-square test value based on the contingency table.

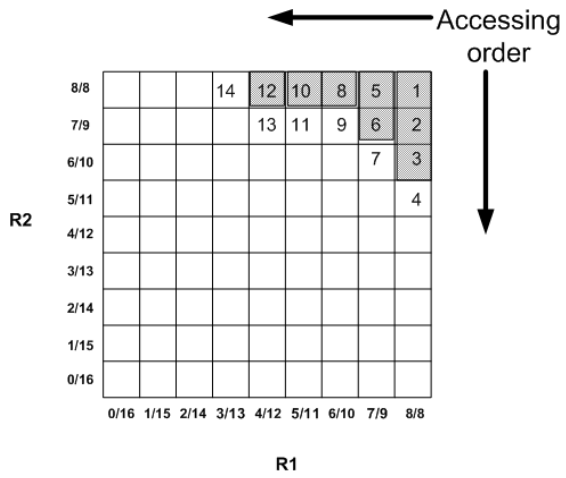


Figure 3.2: Accessing $Array(X_i)$ to retrieve the candidate SNP-pairs

Efficient retrieval of candidate SNP-pairs

To efficiently retrieve the candidates, SNP-pairs (X_iX_j) in $AP(X_i)$ are grouped by their $(\mathscr{R}_1(X_iX_j), \mathscr{R}_2(X_iX_j))$ values and indexed in a 2D array, referred to as $Array(X_i)$.

Example 3.4.4. *Following Example 5.5.2, Figure 3.2 shows the* 9×9 *array,* $Array(X_i)$*, whose entries represent the possible values of* $(\mathscr{R}_1(X_iX_j), \mathscr{R}_2(X_iX_j))$ *for SNP-pairs* $(X_iX_j) \in AP(X_i)$*. The entries in the same column have the same* $\mathscr{R}_1(X_iX_j)$ *value, and their* $\mathscr{R}_2(X_iX_j)$ *values increase from bottom to top. The entries in the same row have the same* $\mathscr{R}_2(X_iX_j)$ *value, and their* $\mathscr{R}_1(X_iX_j)$ *values increase from left to right. The* $\mathscr{R}_1(X_iX_j)$ *value of each column is noted beneath each column. The* $\mathscr{R}_2(X_iX_j)$ *value of each row is noted left to each row. Each entry of the array is a pointer to the SNP-pairs* $(X_iX_j) \in AP(X_i)$ *having the corresponding* $(\mathscr{R}_1(X_iX_j), \mathscr{R}_2(X_iX_j))$ *values.*

In order to find the candidates SNP-pairs whose upper bounds are greater than θ, we start from the right most column of the array, i.e., the entries having the largest $\mathscr{R}_1(X_iX_j)$ value. We scan this column from the top (entries have larger $\mathscr{R}_2(X_iX_j)$ values) towards the bottom (entries have smaller $\mathscr{R}_2(X_iX_j)$ values). If an entry satisfies the inequality $\chi^2(X_i, Y) + T_1S_1\mathscr{R}_1 + T_2S_2\mathscr{R}_2 \geq \theta$, then the SNP-pairs indexed by it are the candidates subject to the two-

locus chi-square tests. Once we reach an entry violating the inequality, we stop searching the current column, since the remaining entries in the column have the same $\mathscr{R}_1(X_iX_j)$ values but smaller $\mathscr{R}_2(X_iX_j)$ values. They will not satisfy the inequality either. Thus all the SNP-pairs indexed by the remaining entries are pruned without further examination. We then move to the column with the next smaller $\mathscr{R}_1(X_iX_j)$ value and repeat the same scanning process again. This whole process terminates when (1) we finish examining all columns or (2) we reach a column whose top entry does not satisfy the inequality. In the latter case, we can safely prune SNP pairs in the remaining columns since none of them can satisfy the inequality.

Example 3.4.5. *Continuing with Examples 5.5.2 and 3.4.4, the entries numbered from 1 to 14 in Figure 3.2 are the ones visited by the scanning process. The numbers show the order in which the entries are visited. Only the SNP-pairs indexed by shaded entries need to be evaluated by chi-square tests. The SNP-pairs indexed by the blank entries, including the entries on the boundary (numbered as* $\{4, 7, 9, 11, 13, 14\}$*) can be safely pruned without examination because their chi-square upper bounds are less than* θ*.*

Property 3.4.6. *For any SNP* X_i*, the maximum number of the entries in* $Array(X_i)$ *is*

$$\lfloor \frac{M}{2} \rfloor \times \lceil \frac{M}{2} \rceil.$$

The proof of Property 3.4.6 is straightforward and omitted here. Note that for typical genome wide association studies, the number of individuals M is much smaller than the number of SNPs N. Therefore, the additional computational cost for accessing $Array(X_i)$ is minimal compared to chi-square tests for all pairs $(X_iX_j) \in AP(X_i)$, even if we examine every entry in this 2D array. In practice, a large portion of the SNP-pairs are pruned. FastChi only needs to access a small number of entries located in the top right corner of $Array(X_i)$ as shown in Figure 3.2. To index the SNPs by the entries of $Array(X_i)$, FastChi scans all $(X_iX_j) \in AP(X_i)$ for their $(\mathscr{R}_1(X_iX_j), \mathscr{R}_2(X_iX_j))$ values. However, as discussed before, this is a much more efficient process than performing two-locus chi-square tests on them.

Algorithm 3: FastChi

Input: SNPs $X' = \{X_1, X_2, \cdots, X_N\}$, phenotype permutations $Y' = \{Y_1, Y_2, \cdots, Y_K\}$, and input parameter θ

Output: for every $Y_k \in Y'$, find the set of SNP-pairs $Result(Y_k) = \{(X_iX_j) | \chi^2(X_iX_j, Y_k) \geq \theta, 1 \leq i < j \leq N\}$

```
for every X_i ∈ X', do
    index (X_iX_j) ∈ AP(X_i) by Array(X_i);
    for every Y_k ∈ Y', do
        access Array(X_i) to find the candidate SNP-pairs and store them in
        Cand(X_i, Y_k);
        for every (X_iX_j) ∈ Cand(X_i, Y_k) do
            if χ²(X_iX_j, Y_k) ≥ θ then
                Result(Y_k) ← (X_iX_j);
            end
        end
    end
end
Return Result(Y_k) for all Y_k ∈ Y'.
```

3.4.2 Permuting the Phenotype

For multiple correlated tests, permutation procedure is often used in genetic analysis for proper family wise errors. For genome association mapping, permutation is less commonly used because it is often entails prohibitively long computation time. Our FastChi algorithm makes permutation procedure more feasible in genome-wide association mapping.

Let $Y' = \{Y_1, Y_2, \cdots, Y_K\}$ be the K permutations of the phenotype Y. Following the idea discussed in Section 3.4.1, the bound established by Inequality (3.7) can be easily incorporated in the algorithm to handle the permutations.

Property 3.4.7. *For every SNP X_i, the indexing structure $Array(X_i)$ is independent of permutations in Y'.*

The correctness of this property relies on the fact that for any $(X_iX_j) \in AP(X_i)$, its $(\mathscr{R}_1(X_iX_j), \mathscr{R}_2(X_iX_j))$ value does not change over different permutations. Thus for each X_i, once we get $Array(X_i)$, it can be reused in all permutations.

The FastChi algorithm is described in Algorithm 4. The inputs of FastChi include the N

SNPs, K phenotype permutations, and threshold θ. For every phenotype, FastChi finds the SNP-pairs whose chi-square test value $\chi^2(X_iX_j, Y_k) \geq \theta$. If no permutation correction is needed, then $K = 1$, and the input phenotype is just the original phenotype Y. For each X_i, FastChi first indexes $(X_iX_j) \in AP(X_i)$ using $Array(X_i)$. Then it finds the set of candidate SNP-pairs $Cand(X_i, Y_k)$ by accessing $Array(X_i)$ for every phenotype permutation Y_k. The candidates in $Cand(X_i, Y_k)$ are then evaluated for their chi-square test values. The candidates whose chi-square test values are greater than or equal to θ are reported by the algorithm.

3.4.3 Complexity Analysis

Time complexity: For each X_i, FastChi needs to index $(X_iX_j) \in AP(X_i)$. The complexity to build the indexing structure for all SNPs is thus $O(N(N-1)M/2)$. The worst case for accessing all $Array(X_i)$ for all permutations is $O(N \times K \times \lfloor \frac{M}{2} \rfloor \times \lceil \frac{M}{2} \rceil)$. Let $\sum_{i,k} |Cand(X_i, Y_k)|$ represent the total number of candidates. The overall time complexity of FastChi is thus $O(N(N-1)M/2) + O(NK \times \lfloor \frac{M}{2} \rfloor \times \lceil \frac{M}{2} \rceil) + O(\sum_{i,k} |Cand(X_i, Y_k)|)$. The experimental results show that the overhead of building the indexing structures and accessing them for candidates are negligible comparing to the time spent on performing chi-square tests for the candidate SNP-pairs when the permutation correction is needed. Note that the time complexity of the brute force approach is $O(KN(N-1)M/2)$.

Space complexity: The total number of variables in the dataset, including the SNPs and the phenotype permutations, is $N+K$. The maximum space of the indexing structure $Array(X_i)$ is $O(\lfloor \frac{M}{2} \rfloor \times \lceil \frac{M}{2} \rceil + N)$. Note that for each X_i, FastChi only needs to access one indexing structure, $Array(X_i)$, for all permutations. Once the evaluation process for X_i is over for all permutations, $Array(X_i)$ can be cleared from the memory. Therefore, the space complexity of FastChi is $O((N+K)M) + O(\lfloor \frac{M}{2} \rfloor \times \lceil \frac{M}{2} \rceil + N)$. Since usually M is much smaller than N, this space complexity is roughly linear to the dataset size.

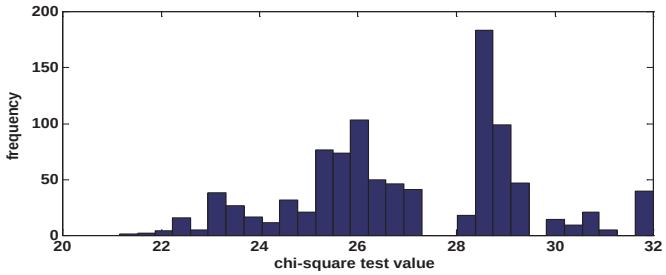


Figure 3.3: Distribution of the maximum chi-square test values of 1000 permutations

3.5 Experimental Results

In this section, we present extensive experimental results on evaluating the performance of the FastChi algorithm. We show (1) the runtime comparison between FastChi and the brute force approach under various experimental settings, (2) the punning effect of the upper bound, and (3) the relative computational cost of each component of FastChi, including building the indexing structures for the SNP-pairs, accessing them to find the candidate SNP-pairs, and performing chi-square tests on the candidates. FastChi is implemented in C++. The experiments are performed on a 2.4 GHz PC with 1G memory running WindowsXP system.

Dataset: The SNP dataset used for the experiments is extracted from a set of combined SNPs from the 140k Broad/MIT mouse dataset and 10k GNF mouse dataset. This merged dataset has 156,525 SNPs for 71 mouse strains. The missing values in the dataset are imputed using NPUTE (Roberts et al. (2007)).

Experimental settings: The phenotypes used in the experiments are random permutations of binary variable which contains half cases and half controls. This is common in practice, where the numbers of cases and controls tend to be balanced. If not otherwise specified, the default setting of the experiments are as follows: #individuals = 32, #SNPs=8k, #permutations=20. There are 60,970 unique SNPs for these 32 mice strains. To find the appropriate threshold value, we permute the phenotypes 1000 times. Figure 3.3 shows the distribution of the maximum chi-square test values of the 1000 permutations. Using a critical significance level of 1%, we set the default threshold value of θ to be 32 for the experiments.

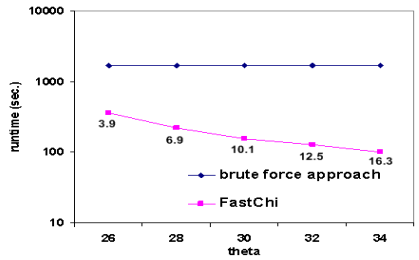
Note that these experimental settings are chosen to demonstrate the performance gain and enhanced scalability offered by FastChi over the brute force implementation of two-locus chi-square test. In real utility, one may use larger SNP panels and/or larger number of permutation tests. The performance of FastChi is expected to follow the same trends presented in the remainder of this section.

3.5.1 FastChi v.s. the brute force approach

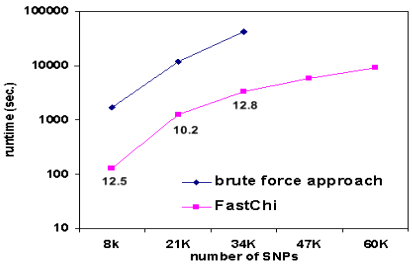
In this subsection, we compare the runtime of FastChi and the brute force approach under various experimental settings. The implementation of the brute force approach includes the computation of two-locus chi-square test for every SNP pairs. Figures 3.4(a) 3.4(b) 3.4(c) 3.4(d) show the running time comparison of FastChi and the brute force approach under varying parameter settings. The y-axis is in logarithm scale. The numbers below the runtime line of FastChi indicate the ratio of the runtime of the brute force approach and the runtime of FastChi.

Figure 3.4(a) shows the runtime comparison between FastChi and the brute force approach under a wide range of threshold values. The runtime of the brute force approach does not change over different θ values. However, the runtime of FastChi dramatically decreases as θ increases. FastChi offers 3.9 fold speedup when $\theta = 26$ and 16.3 fold speedup when $\theta = 34$. This is consistent with the result on the pruning effect of the upper bound, which will be further discussed in Section 3.5.2. In general, the higher the value of θ, the more powerful the pruning effect, hence the faster the algorithm.

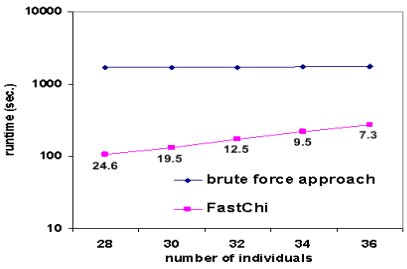
Figure 3.4(b) and Figure 3.4(c) show the runtime of FastChi and the brute force approach when the size of the dataset varies. Figure 3.4(b) depicts the comparison of these two approaches when the number of unique SNPs changes. From this figure, it is clear that FastChi is about an order of magnitude faster than the brute force approach. The brute force approach cannot finish after 15 hours running when the number of unique SNPs is greater than 40k. Thus for the brute force approach, the results for 47k and 60k SNPs are not shown in the



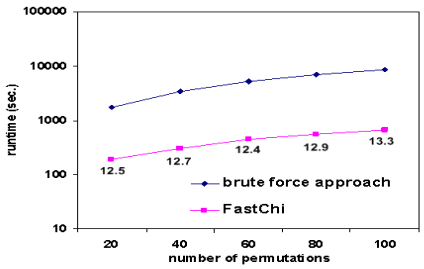
(a) Varying θ



(b) Varying #SNPs



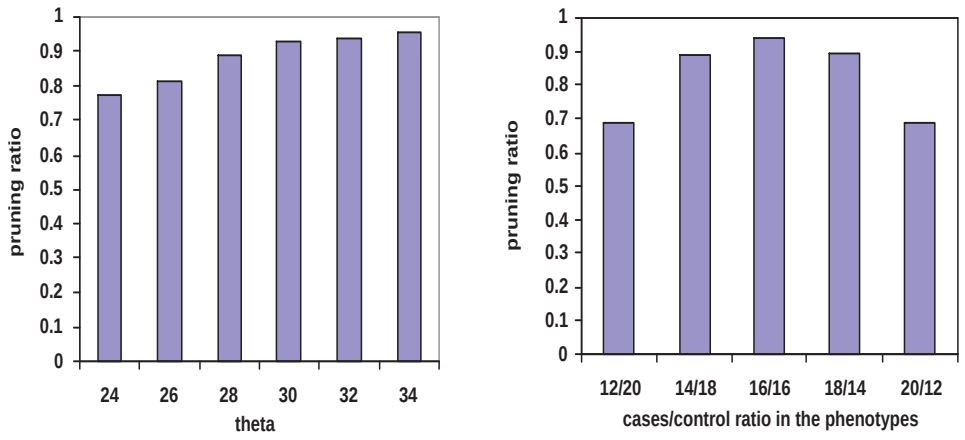
(c) Varying #individuals



(d) Varying #permutations

Figure 3.4: Performance comparisons between FastChi and the brute force approach under different settings.

figure. Figure 3.4(c) shows the comparison of these two approaches when the number of individuals in the dataset changes. The runtime of Fastchi increases as the number of individuals



(a) Varying threshold values (b) Varing case/control ratios

Figure 3.5: Pruning effect of the upper bound

increases. This is because more SNPs-pairs will have larger chi-square values when the number of individuals increases. Their upper bounds will also increase accordingly. In practice, it is reasonable to set higher threshold values for the datasets containing more individuals.

Figure 3.4(d) shows the runtime comparison of FastChi and the brute force approach when the number of phenotype permutations changes. Both runtimes are linear with respect to the number of permutations. FastChi is consistently an order of magnitude faster than the brute force approach when the number of permutations varies.

3.5.2 Pruning effect of the upper bound

In this subsection, we examine the pruning power of the upper bound. Figure 3.5(a) shows the ratio of the SNP-pairs pruned and the total number of SNP-pairs under different thresholds. The pruning ratio is averaged on 20 random phenotype permutations. The phenotype permutations contain half cases and half controls. As we can see from the figure, using the upper bound in Inequality (3.7), a large portion of the SNP-pairs are pruned even when the threshold is low.

We further study the pruning effect when the case/control ratio in the phenotype changes.

Figure 3.5(b) show the pruning ratio of the SNP-pairs when the phenotype contains different number of cases, while the total number of cases and controls is fixed to 32. It is clear from the figure that the pruning effect reaches the maximum power if the phenotype contains 16 cases and 16 controls, which demonstrates that our approach is more suitable for the balanced case-control study.

3.5.3 Computational cost of each component of FastChi

We further examine the detailed computational cost of each component of the FastChi algorithm. FastChi has three major components: building the indexing structure $Array(X_i)$ for every SNP X_i, accessing $Array(X_i)$ to find the candidate SNP-pairs whose upper bounds are greater or equal to the threshold, and performing chi-square tests on these candidates.

Figure 3.6 shows the runtime of these three components when the number of SNPs in the dataset varies. The three lines from the bottom show the runtimes of these three components. We also plot the runtime of the brute force approach for reference, which is the top line. Note that the runtimes in this figure are for one permutation. As we can see from this figure, the cost for accessing the indexing structures is the minimum one. This demonstrates the efficiency of the method introduced in Section 3.4.1 for candidate retrieval. Performing two-locus chi-square tests on candidate SNP pairs takes less than 1/10 of the time required to perform such tests on all SNP-pairs. This is the result of the upper bound pruning since most SNP-pairs have been pruned and only a small number of candidates need to be evaluated for their chi-square values. Among the three components of FastChi, the most time consuming one is on building the index structures. Yet, its runtime is about 1/5 of the time required to perform the two-locus chi-square tests on all SNP pairs in one permutation. When the number of permutation is large, the cost on building the index structures is negligible since they only need to be built once and can be reused in all permutations. Thus the performance gain of FastChi is expected to be more prominent for large permutation tests.

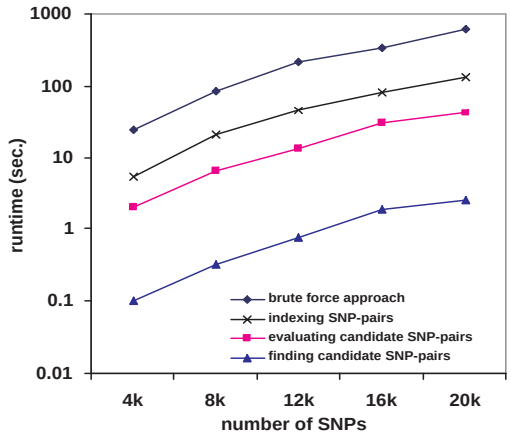


Figure 3.6: Computational cost of each component of FastChi

3.6 Conclusion

In this chapter, we present the FastChi algorithm for genome-wide two-locus chi-square test. FastChi is an exhaustive method which guarantees to find the optimal solution. The efficiency of FastChi is gained from two sources. First, it utilizes an upper bound of the two-locus chi-square test value to prune a majority of the SNP-pairs. The upper bound developed in this paper can be easily incorporated in the algorithm for SNP-pair pruning and candidates retrieval. Second, it identifies computation units that are independent of the phenotype and hence are invariant in permutation tests. By eliminating redundant computation of these invariant units, FastChi is even more effective than the brute force method in permutation test. Extensive experimental results show that FastChi is much more efficient than the brute force alternative.

Chapter 4

The COE Algorithm

4.1 Introduction

In previous chapters, we introduced two algorithms for genome-wide two-locus epistasis detection: FastANOVA for two-locus ANOVA (analysis of variance) test on quantitative traits and FastChi for two-locus chi-square test on case-control phenotypes. Both methods rework the formula of ANOVA test and Chi-square test to estimate an upper bound of the test value for SNP pairs. These upper bounds are used to identify candidate SNP pairs that may have strong epistatic effect. Repetitive computation in a permutation test is also identified and performed once whose results are stored for use by all permutations. These two strategies lead to substantial speedup, especially for large permutation test, without compromising the accuracy of the test. These approaches guarantee to find the optimal solutions. However, a common drawback of these methods is that they are designed for specific tests, i.e., chi-square test and ANOVA test. The upper bounds used in these methods do not work for other statistical tests, such as chi-square test, G-test, information-theoretic association measurements, and trend test (Balding (2006); Pagano and Gauvreau (2000); Thomas (2004)), which are also routinely used by researchers. In addition, new statistics for epistasis detection are continually emerging in the literature (Bohringer et al. (2003); Dong and et al. (2008); Zhao et al. (2005)). Therefore, it is desirable to develop a general model that supports a variety of statistical tests.

In this chapter, we propose a general approach, COE[1], to scale-up the process of genome-wide two-locus epistasis detection. Our method is guaranteed to find the optimal solution. A significant improvement over previous methods is that our approach can be applied to a wide range of commonly used statistical tests. We show that a key property of these statistics is that they are all convex functions of the observed values of certain events in two-locus tests. This allows us to apply the convex optimization techniques (Boyd and Vandenberghe (2004)). Specifically, by examining the contingency tables, we can derive constraints on these observed values. Utilizing these constraints, an upper bound can be derived for the two-locus test value. Similar to the approaches in FastANOVA and FastChi, this upper bound only depends on single-locus test and the genotype of the SNP-pairs. It avoids redundant computation in permutation test by grouping and indexing the SNP-pairs by their genotypes. An important difference, however, is that the upper bound presented in this chapter is general and much tighter than those in previous methods such as FastChi. It supports all tests using convex statistics and can prune the search space more efficiently. As a result, our method is orders of magnitude faster than the brute force approach, in which all SNP-pairs need to be evaluated for their test values, and is an order of magnitude faster than the pruning strategies used in previous methods such as FastChi.

4.2 The Problem

Let $\{X_1, X_2, \cdots, X_N\}$ be the set of all biallelic SNPs for M individuals, and Y be the binary phenotype of interest (e.g., disease or non-disease). We adopt the convention of using 0 to represent majority allele and 1 to represent minority allele, and use 0 for non-disease and 1 for disease. We use $\mathscr{T}$ to denote the statistical test. Specifically, we represent the test value of SNP X_i and phenotype Y as $\mathscr{T}(X_i, Y)$, and represent the test value of SNP-pair (X_iX_j) and Y as $\mathscr{T}(X_iX_j, Y)$. A contingency table, which records the observed values of all events, is

[1]COE stands for Convex Optimization-based Epistasis detection algorithm.

the basis for many statistical tests. Table 4.1 shows contingency tables for the single-locus test $\mathscr{T}(X_i, Y)$, genotype relationship between SNPs X_i and X_j, and two-locus test $\mathscr{T}(X_iX_j, Y)$.

(a) X_i and Y

	$X_i = 0$	$X_i = 1$	Total
$Y = 0$	event A	event B	
$Y = 1$	event C	event D	
Total			M

(b) X_i and X_j

	$X_i = 0$	$X_i = 1$	Total
$X_j = 0$	event S	event T	
$X_j = 1$	event P	event Q	
Total			M

(c) X_iX_j and Y

	$X_i = 0$		$X_i = 1$		Total
	$X_j = 0$	$X_j = 1$	$X_j = 0$	$X_j = 1$	
$Y = 0$	event a_1	event a_2	event b_1	event b_2	
$Y = 1$	event c_1	event c_2	event d_1	event d_2	
Total					M

Table 4.1: Contingency tables

The goal of permutation test is to find a critical threshold value. A two-locus epistasis detection with permutation test is typically conducted as follows Pagano and Gauvreau (2000); Zhang et al. (2008, 2009). A permutation Y_k of Y represents a random reshuffling of the phenotype Y. In each permutation, the phenotype values are randomly reassigned to individuals with no replacement. Let $Y' = \{Y_1, Y_2, \cdots, Y_K\}$ be the set of K permutations of Y. For each permutation $Y_k \in Y'$, let $\mathscr{T}_{Y_k}$ represent the maximum test value among all SNP-pairs, i.e., $\mathscr{T}_{Y_k} = \max\{\mathscr{T}(X_iX_j, Y_k) | 1 \leq i < j \leq N\}$. The distribution of $\{\mathscr{T}_{Y_k} | Y_k \in Y'\}$ is used as the null distribution. Given a Type I error threshold α, the *critical value* $\mathscr{T}_\alpha$ is the αK-th largest value in $\{\mathscr{T}_{Y_k} | Y_k \in Y'\}$. After determining the critical value $\mathscr{T}_\alpha$, a SNP-pair (X_iX_j) is considered significant if its test value with the original phenotype Y exceeds the critical value, i.e., $\mathscr{T}_(X_iX_j, Y) \geq \mathscr{T}_\alpha$.

Determining the critical value is computationally more demanding than finding significant SNP-pairs, since the test procedure needs to be repeated for every permutation in order to find the maximum values. These two problems can be formalized as follows.

Determining Critical Value: For a given Type I error threshold α, find the critical value $\mathscr{T}_\alpha$, which is the αK-th largest value in $\{\mathscr{T}_{Y_k} | Y_k \in Y'\}$.

Finding Significant SNP-pairs: For a given critical value $\mathscr{T}_\alpha$, find the significant SNP-

pairs (X_iX_j) such that $\mathscr{T}(X_iX_j, Y) \geq F_\alpha$.

In the reminder of the chapter, we first show the convexity of common statistics. Then we discuss how to establish an upper bound of two-locus test and use it in the algorithm to efficiently solve the two problems.

4.3 Convexity of Common Test Statistics

In this section, we show that many commonly used statistics are convex functions. Since there are many statistics in the literature, it is impossible to exhaustively enumerate all of them. We focus on four widely used statistics: chi-square test, G-test, entropy-based statistic, and Cochran-Armitage trend test.

Let $A, B, C, D, S, T, P, Q, a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2$ represent the events as shown in Table 4.1. Let E_{event} and O_{event} denote the expected value and observed value of an event. Suppose that $\mathbb{E}_0 = \{a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2\}$, $\mathbb{E}_1 = \{a_1, a_2, c_1, c_2\}$, and $\mathbb{E}_2 = \{b_1, b_2, d_1, d_2\}$. The two-locus chi-square tests can be calculated as follows:

$$\chi^2(X_iX_j, Y) = \underbrace{\sum_{event \in \mathbb{E}_1} \frac{(O_{event} - E_{event})^2}{E_{event}}}_{\chi_1^2(X_iX_jY)} + \underbrace{\sum_{event \in \mathbb{E}_2} \frac{(O_{event} - E_{event})^2}{E_{event}}}_{\chi_2^2(X_iX_jY)}. \tag{4.1}$$

Note that we intentionally break the calculation into two components: one for the events in $\mathbb{E}_1$, denoted as $\chi_1^2(X_iX_jY)$, and one for the events in $\mathbb{E}_2$, denoted as $\chi_2^2(X_iX_jY)$. The reason for separating these two components is that each of these two components is a convex function (See Lemma 4.3.1).

The G-test, also known as a likelihood ratio test for goodness of fit, is an alternative to the chi-square test. The formula for two-locus G-test is

$$G(X_iX_j, Y) = 2 \sum_{event \in \mathbb{E}_1} O_{event} \cdot \ln(\frac{O_{event}}{E_{event}}) + 2 \sum_{event \in \mathbb{E}_2} O_{event} \cdot \ln(\frac{O_{event}}{E_{event}}). \tag{4.2}$$

Information-theoretic measurements have been proposed for association study (Dong and et al. (2008); Zhao et al. (2005)). We examine the mutual information measure, which is the basic form of many other measurements. The mutual information between SNP-pair (X_iX_j) and phenotype Y is $I(Y; X_iX_j) = H(Y)+H(X_iX_j)-H(X_iX_jY)$, in which the joint entropy $-H(X_iX_jY)$ is calculated as

$$-H(X_iX_jY) = \sum_{event\in\mathbb{E}_1} \frac{O_{event}}{M} \cdot \log \frac{O_{event}}{M} + \sum_{event\in\mathbb{E}_2} \frac{O_{event}}{M} \cdot \log \frac{O_{event}}{M}. \tag{4.3}$$

Let $\mathscr{T}(X_iX_j, Y)$ represent any one of $\chi^2(X_iX_j, Y)$, $G(X_iX_j, Y)$, and $-H(X_iX_jY)$. Let $\mathscr{T}_1(X_iX_jY)$ denote the component for events in $\mathbb{E}_1$, and $\mathscr{T}_2(X_iX_jY)$ denote the component for events in $\mathbb{E}_2$. The following lemma shows the convexity of $\mathscr{T}_1(X_iX_jY)$ and $\mathscr{T}_2(X_iX_jY)$.

Lemma 4.3.1. *Given the values of $O_A, O_B, O_C, O_D, O_P, O_Q$, $\mathscr{T}_1(X_iX_jY)$ is a convex function of O_{c_2}, and $\mathscr{T}_2(X_iX_jY)$ is a convex function of O_{d_2}.*

Proof. We first show that $\chi^2_1(X_iX_j, Y)$ is a convex function of O_{c_2}. Recall that

$$\chi^2_1(X_iX_j, Y) = \sum_{event\in\{a_1,a_2,c_1,c_2\}} \frac{(O_{event} - E_{event})^2}{E_{event}}.$$

For fixed $O_A, O_B, O_C, O_D, O_P, O_Q$, we know that the expected values of the four events are constants:

$$\begin{cases} E_{a_1} = \dfrac{O_S(O_A + O_B)}{M} = \dfrac{(O_A + O_C - O_P)(O_A + O_B)}{M} \\ E_{a_2} = \dfrac{O_P(O_A + O_B)}{M} \\ E_{c_1} = \dfrac{O_S(O_C + O_D)}{M} = \dfrac{(O_A + O_C - O_P)(O_C + O_D)}{M} \\ E_{c_2} = \dfrac{O_P(O_C + O_D)}{M} \end{cases}$$

From the relations between the observed values of the events in two-locus test, we have that O_{a_1}, O_{a_2}, O_{c_1} are linear functions of O_{c_2}. So $\chi^2_1(X_iX_j, Y)$ is a positive quadratic function of O_{c_2}. Thus $\chi^2_1(X_iX_j, Y)$ is a convex function of O_{c_2}.

Next, we show that

$$G_1(X_iX_j, Y) = \sum_{event \in \{a_1, a_2, c_1, c_2\}} O_{event} \cdot \ln \frac{O_{event}}{E_{event}}$$

is a convex function of O_{c_2}. From previous result, for fixed $O_A, O_B, O_C, O_D, O_P, O_Q$, the expected values of the four events $\{a_1, a_2, c_1, c_2\}$ are constants, and $O_{a_1}, O_{a_2}, O_{c_1}$ are linear functions of O_{c_2}. Thus $G_1(X_iX_j, Y)$ is a function of O_{c_2}. To prove the convexity of $G_1(X_iX_j, Y)$, it suffices to show that the second derivative $\nabla^2 G_1(X_iX_j, Y) = \frac{\partial^2 G_1(X_iX_j, Y)}{\partial O_{c_2}^2}$ is nonnegative. We show this is the case for the component of event a_2:

$$\nabla^2(O_{a_2} \cdot \ln \frac{O_{a_2}}{E_{a_2}}) = \nabla^2((O_P - O_{c_2}) \cdot \ln \frac{O_P - O_{c_2}}{E_{a_2}}) = \frac{1}{O_P - O_{c_2}} \geq 0.$$

Similarly, we can prove that the second derivative of other components are nonnegative. Therefore, $G_1(X_iX_j, Y)$ is a convex function of O_{c_2}.

Following the similar idea, i.e., by showing the second derivative of $-H(X_iX_jY)$ is nonnegative, we can prove that $-H_1(X_iX_jY)$ is a convex function of O_{c_2}.

Thus we have shown the $\mathscr{T}_1(X_iX_jY)$ is a convex function of O_{c_2}. The convexity $\mathscr{T}_2(X_iX_jY)$ can be proven in a similar way. □

The Cochran-Armitage test for trend is another widely used statistic in genetic association study. Let $Z = (O_{c_1} - pO_S)(s_1 - \bar{s}) + (O_{c_2} - pO_P)(s_2 - \bar{s}) + (O_{d_1} - pO_T)(s_3 - \bar{s}) + (O_{d_2} - pO_Q)(s_4 - \bar{s})$. The Cochran-Armitage two-locus test can be calculated as

$$z^2 = Z^2/[p(1-p)(O_S(s_1 - \bar{s})^2 + O_P(s_2 - \bar{s})^2 + O_T(s_3 - \bar{s})^2 + O_Q(s_4 - \bar{s})^2)],$$

where p is the percentage of cases in the case-control population, s_i ($i \in \{1, 2, 3, 4\}$) are user specified scores for the four possible genotype combinations of (X_iX_j): $\{00, 01, 10, 11\}$, and $\bar{s} = (O_S s_1 + O_P s_2 + O_T s_3 + O_Q s_4)/M$ is the weighted average score. The following theorem shows the convexity of the trend test.

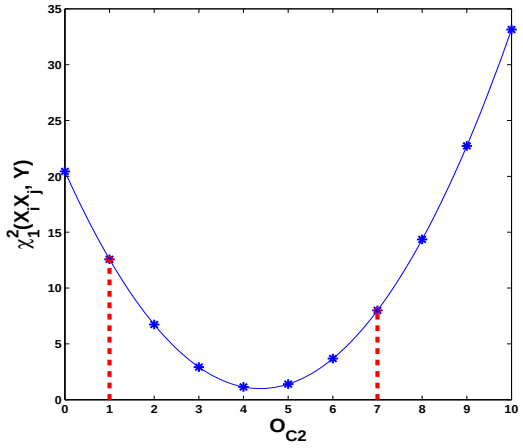


Figure 4.1: Convexity Example

Lemma 4.3.2. *Given the values of $O_A, O_B, O_C, O_D, O_P, O_Q$, the Cochran-Armitage test for trend z^2 is a convex function of (O_{c_2}, O_{d_2}).*

Proof. Observe that the O_{c_1} is a linear function of O_{c_2}, and O_{d_1} is a linear function of O_{d_2}. The values of p, s_i ($i \in \{1, 2, 3, 4\}$), and $\bar{s}$ are fixed. Thus the trend statistic z^2 is a quadratic function of the two variables (O_{c_2}, O_{d_2}). This completes the proof. □

Suppose that the range of O_{c_2} is $[l_{c_2}, u_{c_2}]$, and the range of O_{d_2} is $[l_{d_2}, u_{d_2}]$. For any convex function, its maximum value is attained at one of the vertices of it convex domain (Boyd and Vandenberghe (2004)). Thus, from Lemmas 4.3.1 and 4.3.2, we have the following theorem.

Theorem 4.3.3. *Given the values of $O_A, O_B, O_C, O_D, O_P, O_Q$, for chi-square test, G-test, and entropy-based test, the maximum value of $\mathscr{T}_1(X_iX_jY)$ is attained when $O_{c_2} = l_{c_2}$ or $O_{c_2} = u_{c_2}$. The maximum value of $\mathscr{T}_2(X_iX_jY)$ is attained when $O_{d_2} = l_{d_2}$ or $O_{d_2} = u_{d_2}$. The maximum value of Cochran-Armitage test z^2 is attained when (O_{c_2}, O_{d_2}) takes one of the four values in $\{(l_{c_2}, l_{d_2}), (l_{c_2}, u_{d_2}), (u_{c_2}, l_{d_2}), (u_{c_2}, u_{d_2})\}$.*

Therefore, we can develop an upper bound of the two-locus test if we identify the range of O_{c_2} and O_{d_2}. For example, suppose that the value of vector $(O_A, O_B, O_C, O_D, O_P, O_Q)$ is $(6, 10, 10, 6, 7, 6)$. In Figure 4.1, we plot function $\chi^2_1(X_iX_j, Y)$. The blue stars represent the

values of $\chi_1^2(X_iX_j, Y)$ when O_{c_2} takes different values. Clearly, $\chi_1^2(X_iX_j, Y)$ is a convex function of O_{c_2} , and its upper bound is determined by the two end points of the range of O_{c_2}. Since O_{c_2} is always less than O_C, in this example, the default range of O_{c_2} is $[0, O_C] = [0, 10]$. Typically, the actual range of O_{c_2} is tighter, as indicated by the red dotted lines, which leads to a tighter upper bound of the test value. In the next section, by examining the contingency tables, we derive a set of constraints that determine the range of O_{c_2} and O_{d_2}.

4.4 Constraints on Observed Values

$$\begin{cases} O_{a_1} + O_{a_2} = O_A \\ O_{b_1} + O_{b_2} = O_B \\ O_{c_1} + O_{c_2} = O_C \\ O_{d_1} + O_{d_2} = O_D \\ O_{a_1} + O_{c_1} = O_S \\ O_{a_2} + O_{c_2} = O_P \\ O_{b_1} + O_{d_1} = O_T \\ O_{b_2} + O_{d_2} = O_Q \end{cases} \Longrightarrow \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} O_{a_1} \\ O_{a_2} \\ O_{b_1} \\ O_{b_2} \\ O_{c_1} \\ O_{c_2} \\ O_{d_1} \\ O_{d_2} \end{pmatrix} = \begin{pmatrix} O_A \\ O_C \\ O_B \\ O_D \\ O_P \\ O_Q \end{pmatrix}$$

Figure 4.2: Linear equation system derived from contingency tables

$$\begin{pmatrix} O_{a_1} \\ O_{a_2} \\ O_{c_1} \\ O_{c_2} \end{pmatrix} = \begin{pmatrix} O_A - O_P \\ O_P \\ O_C \\ 0 \end{pmatrix} - \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \end{pmatrix} O_{c_2} \text{ , and } \begin{pmatrix} O_{b_1} \\ O_{b_2} \\ O_{d_1} \\ O_{d_2} \end{pmatrix} = \begin{pmatrix} O_B - O_Q \\ O_Q \\ O_D \\ 0 \end{pmatrix} - \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \end{pmatrix} O_{d_2}.$$

Figure 4.3: Relations between observed values in the contingency table of two-locus test

From the contingency tables shown in Table 4.1, we can develop a set of equations, as shown in Figure 4.2 at the left side of the arrow sign. Although there are 8 equations, the rank of the linear equation system is 6. We choose 6 linear equations to form a full rank system. The matrix multiplication form of these 6 equations is shown in Figure 4.2 at the right side of the arrow sign. The reason for choosing the 6 equations is two-fold. First, these 6 equations

can be used to derive the range of O_{c_2} and O_{d_2}. Second, the values of O_A, O_B, O_C, O_D are determined by the single-locus contingency table in Table 4.1(a). The remaining two values, O_P and O_Q, only depend on the SNP-pair's genotype. It enables us to index the SNP-pairs by their (O_P, O_Q) values to effectively apply the upper bound.

From these 6 equations, we obtain the relationships between the observed values shown in Figure 4.3. Since all observed values in the contingency table must be greater or equal to 0, the ranges of O_{c_2} and O_{d_2} are stated in Theorem 5.5.1.

Theorem 4.4.1. *Given the values of $O_A, O_B, O_C, O_D, O_P, O_Q$, the ranges of O_{c_2} and O_{d_2} are*

$$\begin{cases} \max\{0, O_P - O_A\} \leq O_{c_2} \leq \min\{O_P, O_C\}; \\ \max\{0, O_Q - O_B\} \leq O_{d_2} \leq \min\{O_Q, O_D\}. \end{cases}$$

Given $O_A, O_B, O_C, O_D, O_P, O_Q$, the values of $O_{a_1}, O_{a_2}, O_{c_1}$ are determined by O_{c_2}, the values of $O_{b_1}, O_{b_2}, O_{d_1}$ are determined by O_{d_2}. So all values in the contingency table for two-locus test in Table 4.1(c) depend only on O_{c_2} and O_{d_2}. The maximum value, $ub(\mathscr{T}(X_iX_j, Y))$, is attained when O_{c_2} and O_{d_2} take the boundary values shown in Theorems 5.3.1 and 5.5.1[2]. Continuing with the example in Figure 4.1, the value of $(O_A, O_B, O_C, O_D, O_P, O_Q)$ is $(6, 10, 10, 6, 7, 6)$. From Theorem 5.5.1, the range of O_{c_2} is $[1, 7]$, as indicated by the red lines. The upper bound of $\chi^2_1(X_iX_j, Y)$ is reached when $O_{c_2} = 1$.

Note that the upper bound value only depends on $O_A, O_B, O_C, O_D, O_P, O_Q$. This property allows us to group and index SNP-pairs by their genotypes so that the upper bound can effectively estimated and applied to prune the search space.

[2]For entropy-based statistic, so far we have focused on the joint entropy $-H(X_iX_jY)$. Note that, given the values of $O_A, O_B, O_C, O_D, O_P, O_Q$, the upper bound for the mutual information $I(X_iX_j, Y)$ can also be easily derived.

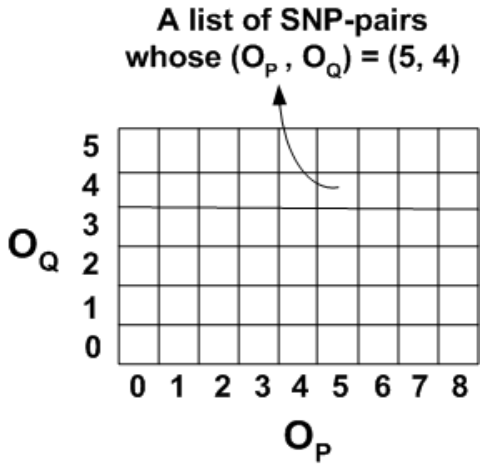


Figure 4.4: Indexing SNP-pairs

4.5 Applying the Upper Bound

Theorems 5.3.1 and 5.5.1 show that the upper bound value of the two-locus test $\mathscr{T}(X_iX_j, Y)$ (for any one of the four tests discussed in Section 4.3) is determined by the values of $O_A, O_B, O_C, O_D, O_P, O_Q$. As shown in Table 4.1, these values only depend on the contingency table for the single-locus test $\mathscr{T}(X_i, Y)$ and the contingency table for the SNP-pair (X_iX_j)'s genotype. This allows us to group the SNP-pairs and index them by their genotypes. The idea of building such indexing structure has also been explored in FastANOVA and FastChi. For completeness, in this section, we first discuss how to apply the upper bound to find the significant SNP-pairs. Then we show that a similar idea can be used to find the critical values $\mathscr{T}_\alpha$ using permutation test.

For every X_i $(1 \leq i \leq N)$, let $AP(X_i) = \{(X_iX_j)|i+1 \leq j \leq N\}$ be the SNP-pairs with X_i being the SNP of lower index value. We can index the SNP-pairs in $AP(X_i)$ by their (O_P, O_Q) values in a 2D array, referred to as $Array(X_i)$. Note that O_P is the number of 1's in X_j when X_i takes value 0. O_Q is the number of 1's in X_j when X_i takes value 1.

For example, suppose that there are 13 individuals in the dataset. SNP X_i consists of 8 0's and 5 1's. Thus for the SNP-pairs in $AP(X_i)$, the possible values of O_P are $\{0, 1, 2, \cdots, 8\}$. The possible values of O_Q are $\{0, 1, 2, \cdots, 5\}$. Figure 4.4 shows the 6×9 array, $Array(X_i)$,

whose entries represent the possible values of (O_P, O_Q) for the SNP-pairs (X_iX_j) in $AP(X_i)$. Each entry of the array is a pointer to the SNP-pairs (X_iX_j) having the corresponding (O_P, O_Q) values. For example, all SNP-pairs in $AP(X_i)$ whose (O_P, O_Q) value is (5,4) are indexed by the entry (5,4) in Figure 4.4.

It is obvious that for any SNP-pair $(X_iX_j) \in AP(X_i)$, if the upper bound value of the two-locus test is less than the critical value, i.e., $ub(\mathscr{T}(X_iX_j, Y)) < \mathscr{T}_\alpha$, then this SNP-pair cannot be significant since its actual test value will also be less than the threshold. Only the SNP-pairs whose upper bound values are greater than the threshold need to be evaluated for their test values. We refer to such SNP-pairs as *candidates*.

Recall that from Theorems 5.3.1 and 5.5.1, the upper bound of two-locus test value is a constant for given $O_A, O_B, O_C, O_D, O_P, O_Q$. Given SNP X_i and phenotype Y, the values of O_A, O_B, O_C, O_D are fixed. For SNP-pairs $(X_iX_j) \in AP(X_i)$, once we index them by their (O_P, O_Q) values as shown in Figure 4.4, we can identify the candidate SNP-pairs by accessing the indexing structure: For each entry of the indexing structure, we calculate the upper bound value. If the upper bound value is greater than or equal to the critical value $\mathscr{T}_\alpha$, then all SNP-pairs indexed by this entry are candidates and subject to two-locus tests. The SNP-pairs whose upper bound values are less than the critical value are pruned without any additional test.

Suppose that there are m 1's and $(M - m)$ 0's in SNP X_i. The maximum size of the indexing structure $Array(X_i)$ is $m(M - m)$. Usually, the number of individuals M is much smaller than the number of SNPs N. Therefore, the number of entries in the indexing structure is also much smaller than N. Thus there must be a group of SNP-pairs indexed by the same entry. Since all SNP-pairs indexed by the same entry have the same upper bound value, the indexing structure enables us to calculate the upper bound value for this group of SNP-pairs together.

So far, we have discussed how to use the indexing structure and the upper bound to prune the search space to find significant SNP-pairs for a given critical value $\mathscr{T}_\alpha$. The problem of finding this critical value $\mathscr{T}_\alpha$ is much more time consuming than finding the significant

SNP-pairs since it involves large scale permutation test. The indexing structure $Array(X_i)$ can be easily incorporated in the algorithm for permutation test. The key property is that the indexing structure $Array(X_i)$ is independent of the phenotype. Once $Array(X_i)$ is built, it can be reused in all permutations. Therefore, building the indexing structure $Array(X_i)$ is only a one time cost. The permutation procedure is similar to that of finding significant SNP-pairs. The only difference is that the threshold used to prune the search space is a dynamically updated critical value found by the algorithm so far. The overall procedure of our algorithm COE is similar to that in FastANOVA and FastChi. An important difference is that COE utilizes the convexity of statistical tests and is applicable to all four statistics. We omit the pseudo code of the algorithm here.

Property 4.5.1. *The indexing structure $Array(X_i)$ can be applied in computing the upper bound value for all four statistical tests, i.e., chi-square test, G-test, mutual information, and trend test.*

The correctness of Property 4.5.1 relies on the fact that the upper bound is always a function of O_A, O_B, O_C, O_D, O_P, O_Q, regardless of the choice of test. All SNP-pairs having the same (O_P, O_Q) value will always share a common upper bound. This property shows that there is no need to rebuild the indexing structure if the users want to switch between different tests. It only needs to be built once and retrieved for later use.

The time complexity of COE for permutation test is $O(N^2M + KNM^2 + CM)$, where N is the number of SNPs, M is the number of individuals, K is the number of permutations, and C is the number of candidates reported by the algorithm. Experimental results show that C is only a very small portion of all SNP-pairs. A brute force approach has time complexity $O(KN^2M)$. Note that N is the dominant factor, since $M \ll N$. The space complexity of COE is linear to the size of the dataset. The derivation of the complexity is similar to that shown for FastANOVA and FastChi.

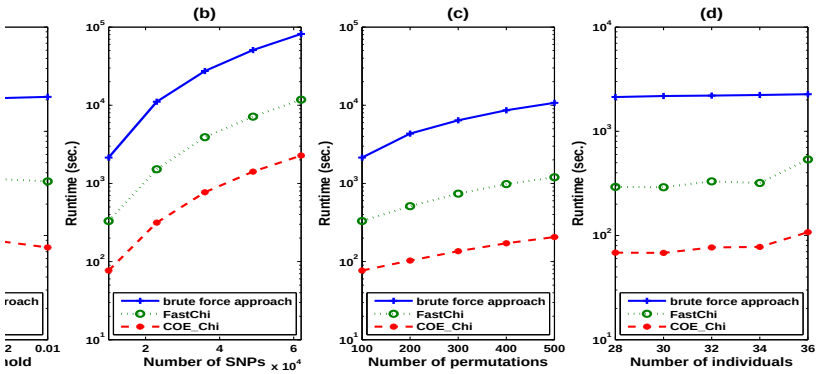


Figure 4.5: Performance comparison of the brute force approach, FastChi, and COE_Chi

4.6 Experimental Results

In this section, we present extensive experimental results on evaluating the performance of the COE algorithm. COE is implemented in C++. We use COE_Chi, COE_G, COE_MI, COE_T to represent the COE implementation for the chi-square test, G-test, mutual information, and trend test respectively. The experiments are performed on a 2.4 GHz PC with 1G memory running WindowsXP system.

Dataset and Experimental Settings: The SNP dataset is extracted from a set of combined SNPs from the 140k Broad/MIT mouse dataset and 10k GNF mouse dataset. This merged dataset has 156,525 SNPs for 71 mouse strains. The missing values in the dataset are imputed using NPUTE (Roberts et al. (2007)). The phenotypes used in the experiments are simulated binary variables which contain half cases and half controls. This is common in practice, where the numbers of cases and controls tend to be balanced. If not otherwise specified, the default settings of the experiments are as follows: #individuals = 32, #SNPs=10,000, #permutations=100. There are 62,876 unique SNPs for these 32 strains.

4.6.1 Performance Comparison

Figure 4.5 shows the runtime comparison of the brute force two-locus chi-square test, the FastChi algorithm Zhang et al. (2009), and the COE implementation of chi-square test, COE_Chi,

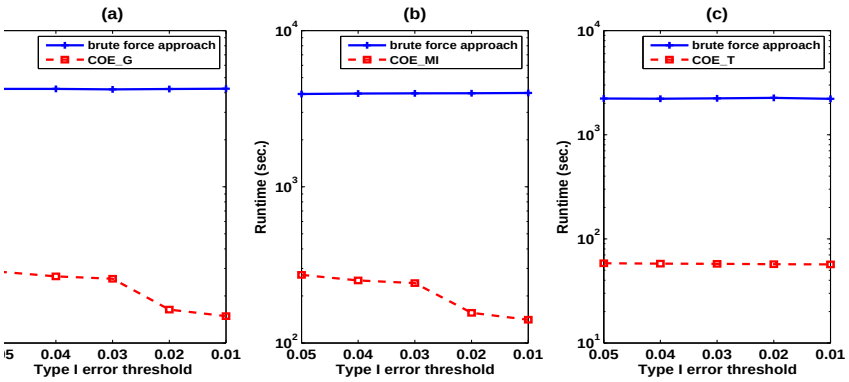


Figure 4.6: Performance comparison of the brute force approach, COE_G, COE_MI, and COE_T

		FastChi	COE_Chi	COE_G	COE_MI	COE_T
α	0.05	87.59%	95.70%	95.84%	95.80%	99.90%
	0.04	87.98%	96.11%	96.23%	96.23%	99.92%
	0.03	88.12%	96.32%	96.40%	96.43%	99.93%
	0.02	89.43%	98.18%	98.31%	98.28%	99.96%
	0.01	90.03%	98.59%	98.65%	98.62%	99.98%
# SNPs	10k	90.03%	98.59%	98.65%	98.62%	99.98%
	23k	91.52%	99.08%	99.50%	99.13%	99.99%
	36k	91.39%	99.03%	99.43%	99.09%	99.99%
	49k	91.39%	99.04%	99.43%	99.09%	99.99%
	62k	91.22%	99.04%	99.43%	99.09%	99.99%
# Perm.	100	90.03%	98.59%	98.65%	98.62%	99.98%
	200	91.79%	99.03%	99.42%	99.08%	99.99%
	300	91.90%	99.04%	99.43%	99.09%	99.99%
	400	91.91%	99.04%	99.43%	99.09%	99.99%
	500	91.99%	99.04%	99.43%	99.09%	99.99%
# Indiv.	28	91.05%	98.77%	99.83%	99.06%	99.99%
	30	91.23%	98.83%	98.94%	99.06%	99.98%
	32	90.03%	98.59%	99.65%	98.62%	99.98%
	34	91.54%	98.80%	99.74%	98.84%	99.97%
	36	89.08%	97.94%	95.74%	93.55%	99.94%

Table 4.2: Pruning effects of FastChi and COE using four different statistics

in permutation test under various settings. Note that the runtime reported in this section are based on the complete executions of all methods including the one time cost for building the indexing structures. Figure 4.5(a) shows the comparison when the Type I error threshold varies. The y-axis is in logarithm scale. COE_Chi improves the efficiency of two-locus epis-

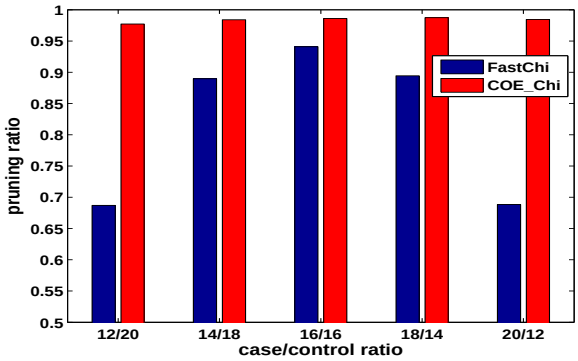


Figure 4.7: FastChi v.s. COE_Chi

tasis detection by one order of magnitude over FastChi (which was specifically designed for two-locus chi-square test), and two orders of magnitude over the brute force approach. Figure 4.5 (b), (c), and (d) demonstrate similar performance improvements of COE_Chi over the other two approaches when varying number of SNPs, number of permutations, and number of individuals respectively. This is consistent with the pruning effect of the upper bounds which will be presented later.

Figure 4.6(a) shows the runtime comparison between the brute force two-locus G-test and COE_G when varying the type I error threshold. The runtime of COE_G dramatically reduces as the type I error threshold decreases. COE_G is one to two orders magnitudes faster than the brute force approach. Similar performance improvement can also be observed for COE_MI and COE_T in Figures 4.6(b) and 4.6(c). Note that for these three tests, we also have similar results when varying other settings.

4.6.2 Pruning Power of the Upper Bound

Table 4.2 shows the percentage of SNP-pairs pruned under different experimental settings for the four statistical tests. We also include the pruning ratio of FastChi in the table for comparison. From the table, most of the SNP-pairs are pruned by COE. Note that COE_Chi has more pruning power than FastChi. The upper bound used in FastChi is derived by loosening the observed values for the events in two-locus test without using the convexity property. The

tighter upper bound of COE_Chi demonstrates the strength of convex optimization in finding the maximum values. In addition, the upper bound derived by applying convex optimization is not only more effective, but also more robust for unbalanced datasets.

Figure 4.7 shows the pruning effectiveness of FastChi and COE_Chi when the ratio of case/control varies. It is clear that the pruning power of FastChi is weakened when the case/control ratio becomes unbalanced. Therefore, FastChi is not very effective for unbalanced case-control datasets. In contrast, COE_Chi maintains a steady pruning percentage under different case/control ratios. Thus it remains effective for the unbalanced datasets. Similar behaviors of COE are also observed in the other three statistical tests.

4.7 Conclusion

In this chapter, we present a general approach COE that support genome-wide disease association study with a wide range of statistics composing of convex terms. We use four commonly used statistics as prototypes: chi-square test, G-test, entropy-based test, and Cochran-Armitage trend test. COE guarantees optimal solution and performs two orders of magnitude faster than brute force approaches.

The performance gain is attributed to two main contributions of COE. The first is a tight upper bound estimated using convex optimization. It has much higher pruning power than any upper bounds used in previous methods such as FastChi. As a result, COE_Chi is an order of magnitude faster than FastChi. Moreover, COE serves as a general platform for two-locus epistasis detection, which eliminates the need of designing specific pruning methods for different statistical tests. Recall that any observed value in a two-locus test is a function of O_{c_2} and O_{d_2} for given $O_A, O_B, O_C, O_D, O_P, O_Q$. Let $x = O_{c_2}$ and $y = O_{d_2}$. A wide spectrum of functions of x and y are convex (Boyd and Vandenberghe (2004)), which include all linear and affine functions on x and/or y, exponential terms e^{ax} ($a \in \mathbb{R}$), powers x^a ($a \geq 1$ or $a \leq 0$), negative logarithm $-\log x$, maximum $\max\{x, y\}$. In addition, many operations

preserve convexity. For example, if $f(x, y)$ is a convex function, and $g(x, y)$ is an affine mapping, then $f(g(x, y))$ is also a convex function. Please refer to (Boyd and Vandenberghe (2004)) for further details.

The second source of performance improvement is from indexing SNP-pairs by their genotypes. Applying this indexing structure, we can compute a common upper bound value for each group. The indexing structure is independent of the phenotype permutations and the choice of statistical test . We can eliminate redundant computation in permutation test and provide the flexibility of supporting multiple statistical tests on the fly.

Chapter 5

The TEAM Algorithm

5.1 Introduction

FastANOVA and FastChi are specifically designed for ANOVA test and chi-square test respectively. The COE algorithm is a more general approach that is applicable to all convex tests. Although these methods provide promising alternatives for GWAS, there are two major drawbacks that limit their applicability. First, they are designed for relatively small sample size and only consider homozygous markers (i.e., each SNP can be represented as a $\{0, 1\}$ binary variable). In human studies, however, the sample size is usually large and most SNPs contain heterozygous genotypes and are coded using $\{0, 1, 2\}$. These make existing methods intractable. Second, although the family-wise error rate (FWER) and false discovery rate (FDR) are both widely used for error controlling (Dudoit and van der Laan (2008); Westfall and Young (1993)), previous methods are designed only to control the FWER. From a computational point of view, the difference in the FWER and the FDR controlling is that, to estimate FWER, for each permutation, only the maximum two-locus test value is needed. To estimate the FDR, on the other hand, for each permutation, all two-locus test values must be computed. Please refer to Section 5.2 for further details of the FWER and the FDR controlling.

In this chapter, we propose an exhaustive algorithm, TEAM[1], for efficient epistasis detec-

[1]TEAM stands for Tree-based Epistasis Association Mapping.

tion in human GWAS. TEAM has several advantages over previous methods.

- It supports to both homozygous and heterozygous data.
- By exhaustively computing all two-locus test values in permutation test, it enables both FWER and FDR controlling.
- It is applicable to all statistics based on contingency tables. Previous methods either are designed for specific tests or require the test statistics satisfy certain property.
- Experimental results demonstrate that TEAM is more efficient than existing methods for large sample study.

TEAM incorporates permutation test for proper error controlling. The key idea is to incrementally update the contingency tables of two-locus tests. We show that only four of the eighteen observed frequencies in the contingency table need to be updated to compute the test value. In the algorithm, we build a minimum spanning tree (Cormen et al. (2001)) on the SNPs. The nodes of the tree are SNPs. Each edge represents the genotype difference between the two connected SNPs. This tree structure can be utilized to speed up updating process for the contingency tables. A majority of the individuals are pruned and only a small portion are scanned to update the contingency tables. This is advantageous in human study, which usually involves thousands of individuals. Extensive experimental results demonstrate the efficiency of the TEAM algorithm.

5.2 The Problem

Suppose that the genotype dataset consists of N SNPs $\{X_1, \cdots, X_N\}$ for M individuals $\{S_1, \cdots, S_M\}$. We adopt the convention of using 0 and 2 to represent the homozygous majority and homozygous minority genotype respectively, and 1 to represent the heterozygous case. Let $Y_0 \in \{0, 1\}$ be the phenotype of interest (0 for controls and 1 for cases). Let

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
X_1	0	0	0	1	2	0	2	0	2	0	0	2	0	0	0	2	0	2	1	0	0	2	2	0
X_2	2	2	0	2	0	2	0	2	2	2	2	0	1	0	0	2	0	2	1	0	2	2	2	2
X_3	2	0	0	2	0	2	0	1	2	1	2	2	1	0	2	2	0	2	1	2	2	2	2	2
X_4	0	2	2	0	0	0	2	1	0	2	2	0	0	0	0	0	0	0	1	0	1	2	0	0
X_5	0	2	2	0	0	0	1	1	2	1	2	0	0	0	0	0	0	2	1	0	2	2	0	2
X_6	0	2	2	0	0	0	2	1	0	1	2	0	0	0	0	2	0	2	1	0	2	2	0	0
Y_1	0	1	0	0	1	1	1	0	0	0	1	1	1	1	1	0	1	0	0	1	0	1	0	0
Y_2	0	0	1	0	1	1	1	1	1	1	0	0	1	0	0	1	1	1	0	0	0	1	0	0
Y_3	1	0	1	1	1	1	0	1	1	1	0	0	0	1	0	1	0	0	1	0	0	0	1	0
Y_4	0	1	1	1	1	1	0	0	0	1	0	0	1	0	1	0	0	0	1	1	0	1	1	0
Y_5	1	0	1	1	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	0	0	1	1	1

Table 5.1: An example dataset

$Y' = \{Y_1, \cdots, Y_K\}$ be the set of K permutations of Y_0. In each permutation Y_k, the phenotype labels are randomly reassigned to individuals with no replacement.

Table 5.1 shows an example dataset of SNPs and phenotype permutations. The genotype dataset consists of 6 SNPs $\{X_1, \cdots, X_6\}$ for 24 individuals $\{S_1, \cdots, S_{24}\}$. Individuals $\{S_1, \cdots, S_{12}\}$ are cases and $\{S_{13}, \cdots, S_{24}\}$ are controls. The phenotype is permuted 5 times, i.e., $Y' = \{Y_1, \cdots, Y_5\}$.

Let $\mathscr{T}$ denote the statistical test to be used. Specifically, we represent the test value of SNP X_i and phenotype Y_k $(0 \leq k \leq K)$ as $\mathscr{T}(X_i, Y_k)$, and represent the test value of SNP-pair (X_iX_j) and Y_k as $\mathscr{T}(X_iX_j, Y_k)$. A contingency table, which records the observed values of certain events, is the basis of many statistical tests. Table 5.2 shows contingency tables for the single-locus test $\mathscr{T}(X_i, Y_k)$ and $\mathscr{T}(X_j, Y_k)$, genotype relationship between SNPs X_i and X_j, and two-locus test $\mathscr{T}(X_iX_j, Y_k)$.

Because of the large number of hypotheses being tested, multiple testing problem has received considerable attention in GWAS. Controlling the FWER and FDR are two widely used approaches to control the error rate. The FWER is the probability of having at least one false positive. The FDR is the expected proportion of false positives among rejected hypotheses. Permutation test is the standard way to estimate the null distribution in both approaches. Next, we briefly describe the typical procedures of the FWER and FDR control. For statistical background of these approaches, please refer to Dudoit and van der Laan (2008); Westfall and Young (1993) for details.

The FWER controlling procedure: For each permutation $Y_k \in Y'$, let $\mathscr{T}_{Y_k}$ represent the

(a) Contingency table for $\mathscr{T}(X_i, Y_k)$

	$X_i = 0$	$X_i = 1$	$X_i = 2$	Total
$Y_k = 0$	event A	event B	event E	
$Y_k = 1$	event C	event D	event F	
Total				M

(b) Contingency table for $\mathscr{T}(X_j, Y_k)$

	$X_j = 0$	$X_j = 1$	$X_j = 2$	Total
$Y_k = 0$	event G	event H	event I	
$Y_k = 1$	event J	event L	event O	
Total				M

(c) Contingency table for two SNPs X_i and X_j

	$X_i = 0$	$X_i = 1$	$X_i = 2$	Total
$X_j = 0$	event S	event T	event R	
$X_j = 1$	event P	event Q	event U	
$X_j = 2$	event V	event W	event Z	
Total				M

(d) Contingency table for (X_iX_j) and Y_k

	$X_i = 0$			$X_i = 1$			$X_i = 2$			Total
	$X_j = 0$	$X_j = 1$	$X_j = 2$	$X_j = 0$	$X_j = 1$	$X_j = 2$	$X_j = 0$	$X_j = 1$	$X_j = 2$	
$Y_k = 0$	event a_1	event a_2	event a_3	event b_1	event b_2	event b_3	event e_1	event e_2	event e_3	
$Y_k = 1$	event c_1	event c_2	event c_3	event d_1	event d_2	event d_3	event f_1	event f_2	event f_3	
Total										M

Table 5.2: Contingency tables for single-locus tests $\mathscr{T}(X_i, Y_k)$, $\mathscr{T}(X_j, Y_k)$, genotype relation between (X_i, X_j), and two-locus test $\mathscr{T}(X_iX_j, Y_k)$

maximum test value among all SNP-pairs, i.e., $\mathscr{T}_{Y_k} = \max\{\mathscr{T}(X_iX_j, Y_k)|1 \leq i < j \leq N\}$. The distribution of $\{\mathscr{T}_{Y_k}|Y_k \in Y'\}$ is used as the null distribution. Given an error rate threshold α, the *critical value* $\mathscr{T}_\alpha$ is the αK-th largest value in $\{\mathscr{T}_{Y_k}|Y_k \in Y'\}$. A SNP-pair (X_iX_j) is considered significant if its test value with the original phenotype Y_0 exceeds the critical value, i.e., $\mathscr{T}(X_iX_j, Y_0) \geq \mathscr{T}_\alpha$.

The FDR controlling procedure: Let PV represent the set of the pooled test values of all permutation tests, i.e., $PV = \{\mathscr{T}(X_iX_j, Y_k)|1 \leq i < j \leq N, 1 \leq k \leq K\}$. The *p*-value of test $\mathscr{T}(X_iX_j, Y_0)$ can be calculated as $p(\mathscr{T}(X_iX_j, Y_0)) = |\{t \geq \mathscr{T}(X_iX_j, Y_0)|t \in PV\}|/|PV|$, i.e., the proportion of the values in PV that are no less than $\mathscr{T}(X_iX_j, Y_0)$. Let $p_{(1)} \leq p_{(2)} \cdots \leq p_{(N(N-1)/2)}$ be the ordered *p*-values of the original tests. Let $v = \max\{u : p_{(u)} \leq \frac{u\alpha}{N(N-1)/2}\}$. The classic Benjamini-Hochberg method rejects all hypotheses for which the corresponding p-values are in the set $\{p_{(1)}, p_{(2)}, \cdots, p_{(v)}\}$.

In the FWER controlling, we only need the maximum test value of each permutation. To control the FDR, all test values need to be computed to estimate the p-value of the original tests. Previous algorithms, such as FastChi and COE, prune the SNP-pairs having weak asso-

ciations. Thus they cannot be used to control the FDR. Our algorithm, TEAM, exhaustively computes the test values of all SNP-pairs for every permutation. It can be used for both the FWER and the FDR controlling. In this paper, we mainly focus on the problem of permutation test, since it is the most computationally intensive procedure. Testing SNP-pairs using original phenotype can be treated as a special case of permutation test.

5.3 Free Variables in the Contingency Table of Two-Locus Test

Let E_{event} and O_{event} denote the expected frequency and observed frequency of an event in Table 5.2. Note that each event represents a subset of individuals. For example, event D is a subset of individuals satisfying $(X_i = 1 \wedge Y_k = 1)$, and O_D represents its observed frequency, i.e., $O_D = |D|$. Using the dataset in Table 5.1, consider X_3 and Y_4 (i.e., $i = 3$ and $k = 4$), we have $D = \{S_{10}, S_{13}, S_{19}\}$, and $O_D = 3$.

Many statistics, such as chi-square test and likelihood ratio test are defined as functions of the observed frequencies in contingency tables. For any test $\mathscr{T}$ based on the contingency table, to calculate the two-locus test value $\mathscr{T}(X_iX_j, Y_k)$, one needs all eighteen observed frequencies for the events in the two-locus contingency table shown in Table 5.2(d). The following theorem shows that we only need four of the eighteen values to calculate the two-locus test value given the three contingency tables in Tables 5.2(a), (b), and (c).

Theorem 5.3.1. *For SNPs X_i, X_j, and permutation Y_k, given the observed frequencies in Tables 5.2(a), (b), and (c), specifically, the values of $\{O_D, O_F, O_J, O_L, O_O, O_S, O_P, O_V, O_T, O_Q, O_W, O_R, O_U, O_Z\}$, all of the observed frequencies in Table 5.2(d) can be determined if the values of $\{O_{d_2}, O_{d_3}, O_{f_2}, O_{f_3}\}$ are known.*

Proof. From the four contingency tables shown in Table 5.2, it is easy to get the following linear equation system:

$$
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
O_{a_1} \\ O_{a_2} \\ O_{a_3} \\ O_{b_1} \\ O_{b_2} \\ O_{b_3} \\ O_{c_1} \\ O_{c_2} \\ O_{c_3} \\ O_{d_1} \\ O_{d_2} \\ O_{d_3} \\ O_{e_1} \\ O_{e_2} \\ O_{e_3} \\ O_{f_1} \\ O_{f_2} \\ O_{f_3}
\end{pmatrix}
=
\begin{pmatrix}
O_A \\ O_B \\ O_C \\ O_D \\ O_E \\ O_F \\ O_G \\ O_H \\ O_I \\ O_J \\ O_L \\ O_O \\ O_S \\ O_P \\ O_V \\ O_T \\ O_Q \\ O_W \\ O_R \\ O_U \\ O_Z
\end{pmatrix}
$$

The rank of the above linear system is 14. We thus take 14 rows $\{4, 6, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21\}$, which form a full rank matrix. The row reduced echelon form of this non-redundant linear system is

$$\left(\begin{array}{cccccccccc|cccccccc|l}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & O_S - O_W + O_D + O_F \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & O_P - O_V \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & O_G - O_U \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & O_T - O_D \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & O_Q \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & O_H \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & O_W - O_D - O_F \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & O_V \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & O_U \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & O_D \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & O_R - O_F \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & O_O \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & O_L \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & O_F
\end{array}\right)$$

Thus we have the following solution:

$$\begin{pmatrix} O_{a_1} \\ O_{a_2} \\ O_{a_3} \\ O_{b_1} \\ O_{b_2} \\ O_{b_3} \\ O_{c_1} \\ O_{c_2} \\ O_{c_3} \\ O_{d_1} \\ O_{e_1} \\ O_{e_2} \\ O_{e_3} \\ O_{f_1} \end{pmatrix} = \begin{pmatrix} O_S - O_W + O_D + O_F \\ O_P - O_V \\ O_G - O_U \\ O_T - O_D \\ O_Q \\ O_H \\ O_W - O_D - O_F \\ O_V \\ O_U \\ O_D \\ O_R - O_F \\ O_O \\ O_L \\ O_F \end{pmatrix} - \begin{pmatrix} 1 \\ -1 \\ 0 \\ -1 \\ 1 \\ 0 \\ -1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} O_{d_2} - \begin{pmatrix} 1 \\ 0 \\ -1 \\ -1 \\ 0 \\ 1 \\ -1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} O_{d_3} - \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 1 \end{pmatrix} O_{f_2} - \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 1 \\ 0 \\ -1 \\ 0 \\ 1 \\ 1 \end{pmatrix} O_{f_3}$$

Clearly, only four variables $\{O_{d_2}, O_{d_3}, O_{f_2}, O_{f_3}\}$ are free. Once the values of these free variables are known, the observed frequencies of remaining events in the two-locus contingency table are also known. □

Suppose that we have all the single-locus contingency tables, i.e., Tables 5.2(a) and (b). Given a SNP-pair (X_i, X_j), Table 5.2(c) is fixed. Thus, from Theorem 5.3.1, for permutation Y_k, once we have the values of $\{O_{d_2}, O_{d_3}, O_{f_2}, O_{f_3}\}$, $\mathscr{T}(X_iX_j, Y_k)$ can be calculated accordingly. In the following, we show that these values can be computed incrementally utilizing a minimum spanning tree built on SNPs. We focus on the incremental process for O_{d_2}. The same process can be applied to update O_{d_3}, O_{f_2}, and O_{f_3}. We first discuss how to update O_{d_2} for a specific permutation. Then we show that the procedure can also handle all the permutations in a batch mode.

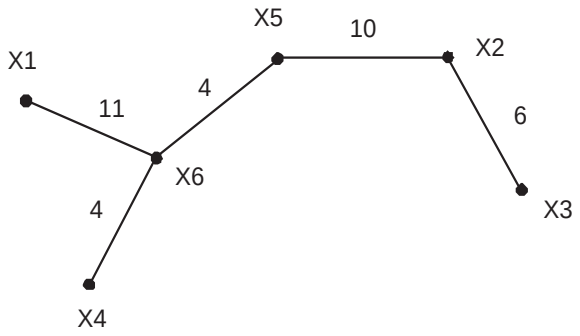


Figure 5.1: The minimum spanning tree built on the SNPs in the example dataset shown in Table 5.1

	$0 \to 1$	$1 \to 0$	$0 \to 2$	$2 \to 0$	$1 \to 2$	$2 \to 1$
(X_3X_2)	$\emptyset$	$\emptyset$	$\{S_2\}$	$\{S_{12}, S_{15}, S_{20}\}$	$\{S_8, S_{10}\}$	$\emptyset$
(X_2X_5)	$\{S_7\}$	$\{S_{13}\}$	$\{S_3\}$	$\{S_1, S_4, S_6, S_{16}, S_{23}\}$	$\emptyset$	$\{S_8, S_{10}\}$
(X_5X_6)	$\emptyset$	$\emptyset$	$\{S_{16}\}$	$\{S_9, S_{24}\}$	$\{S_7\}$	$\emptyset$
(X_6X_1)	$\{S_4\}$	$\{S_8, S_{10}\}$	$\{S_5, S_9, S_{12}, S_{23}\}$	$\{S_2, S_3, S_{11}, S_{21}\}$	$\emptyset$	$\emptyset$
(X_6X_4)	$\emptyset$	$\emptyset$	$\emptyset$	$\{S_{16}, S_{18}\}$	$\{S_{10}\}$	$\{S_{21}\}$

Table 5.3: Genotype difference between the connected SNPs in the minimum spanning tree shown in Figure 5.1

5.4 Building the Minimum Spanning Tree on the SNPs

To build a minimum spanning tree Cormen et al. (2001) on the SNPs, let the SNPs $\{X_1, X_2, \cdots, X_N\}$ be the nodes and SNP-pairs (X_iX_j) $(i \neq j)$ be the (undirected) edges. For each edge (X_iX_j), we denote its weight (the number of individuals having different genotypes in the two SNPs) as $w(X_iX_j)$. A *spanning tree* $\mathcal{T}$ is a tree that spans (connects) all SNPs. Let $V(\mathcal{T})$ be its node set and $E(\mathcal{T})$ be its edge set. A *minimum spanning tree* is a spanning tree whose weight $W_{\mathcal{T}} = \sum w(X_iX_j)$, where $(X_iX_j) \in E(\mathcal{T})$, is no greater than any other spanning tree. Figure 5.1 shows the minimum spanning tree built using the example dataset in Table 5.1. The number on each edge represents its weight. For example, in X_3 and X_2, there are 6 individuals, $\{S_2, S_8, S_{10}, S_{12}, S_{15}, S_{20}\}$, having different genotypes.

For any individual, the genotype difference from X_i to X_j can be any one of the six combinations, i.e., $0 \to 1$ (indicating that the genotype in X_i is 0, and the genotype in X_j is 1), $1 \to 0$, $0 \to 2$, $2 \to 0$, $1 \to 2$, and $2 \to 1$. Using the example dataset in Table 5.1, Table 5.3 shows the genotype differences between the connected two SNPs in the minimum spanning

(a) $D_K(X_3)$ without empty entries

individual id.	phenotype permutations
S_8	$\{Y_2, Y_3\}$
S_{10}	$\{Y_2, Y_3, Y_4, Y_5\}$
S_{13}	$\{Y_1, Y_2, Y_4, Y_5\}$
S_{19}	$\{Y_3, Y_4\}$

(b) Updating $O_{d_2}(X_3X_5)$ from $O_{d_2}(X_3X_2)$

	Y_1	Y_2	Y_3	Y_4	Y_5
$O_{d_2}(X_3X_5)$ after initializing	1	1	1	2	1
$O_{d_2}(X_3X_5)$ after updating for S_1	1	1	1	2	1
$O_{d_2}(X_3X_5)$ after updating for S_8	1	2	2	2	1
$O_{d_2}(X_3X_5)$ after updating for S_{10}	1	3	3	3	2
$O_{d_2}(X_3X_5)$ after updating for S_{13}	0	2	3	2	1

Table 5.4: Updating $O_{d_2}(X_3X_5)$ from $O_{d_2}(X_3X_2)$ for all permutations in a batch mode

tree in Figure 5.1. We use $(X_iX_j)_{\{u\to v\}}$ $(u, v \in \{0, 1, 2\})$ to represent the set of individuals whose genotype in X_i is u and genotype in X_j is v. For example, $(X_3X_2)_{\{1\to 2\}} = \{S_8, S_{10}\}$, and $(X_3X_2)_{\{1\to 2\}\cup\{0\to 2\}} = \{S_2, S_8, S_{10}\}$.

5.5 Incrementally Updating Observed Frequency

In this section, we discuss how to update O_{d_2} by utilizing the minimum spanning tree. For clarity, from now on, we use $d_2(X_iX_j, Y_k)$ to denote the specific event d_2 for the SNP-pair (X_iX_j) and permutation Y_k, i.e., the subsets of individuals satisfying $(X_i = 1 \wedge X_j = 1 \wedge Y_k = 1)$. We use $O_{d_2}(X_iX_j, Y_k)$ to represent its observed frequency, i.e., $O_{d_2}(X_iX_j, Y_k) = |d_2(X_iX_j, Y_k)|$. This notation also applies to other events in the contingency tables shown in Table 5.2. For example, $D(X_i, Y_k)$ represents the subset of individuals satisfying $(X_i = 1 \wedge Y_k = 1)$, and $O_D(X_i, Y_k) = |D(X_i, Y_k)|$.

Next we show that for any SNP-pair (X_iX_j) and an edge $(X_jX'_j) \in E(\mathcal{T})$, given $O_{d_2}(X_iX_j, Y_k)$, how to update the value for $O_{d_2}(X_iX'_j, Y_k)$. From the contingency tables in Table 5.2, it is easy to see that

$$O_{d_2}(X_iX_j, Y_k) = |D(X_i, Y_k) \cap Q(X_i, X_j)|,$$

and

$$O_{d_2}(X_iX'_j, Y_k) = |D(X_i, Y_k) \cap Q(X_i, X'_j)|.$$

The following theorem shows that, given $O_{d_2}(X_iX_j, Y_k)$ and $D(X_i, Y_k)$, using the genotype

difference associated with edge $(X_jX'_j)$, we can get the value of $O_{d_2}(X_iX'_j, Y_k)$.

Theorem 5.5.1. *For any SNP-pair (X_iX_j) and an edge $(X_jX'_j) \in E(\mathcal{T})$, we have $O_{d_2}(X_iX'_j, Y_k) = O_{d_2}(X_iX_j, Y_k) + |D(X_i, Y_k) \cap (X_jX'_j)_{\{0\to1\}\cup\{2\to1\}}|$- $|D(X_i, Y_k) \cap (X_jX'_j)_{\{1\to0\}\cup\{1\to2\}}|$.*

Proof. It suffices to show that

$$D(X_i, Y_k) \cap Q(X_i, X'_j)$$

$$= [D(X_i, Y_k) \cap Q(X_i, X_j)] \cup [D(X_i, Y_k) \cap ((X_jX'_j)_{\{0\to1\}\cup\{2\to1\}})] - [D(X_i, Y_k) \cap ((X_jX'_j)_{\{1\to0\}\{1\to2\}})].$$

This is the same as to show that

$$Q(X_i, X'_j) = Q(X_i, X_j) \cup ((X_jX'_j)_{\{0\to1\}\cup\{2\to1\}}) - ((X_jX'_j)_{\{1\to0\}\{1\to2\}}).$$

This is clearly true, hence completes the proof. □

Example 5.5.2. *Using the example dataset in Table 5.1, let $i = 3$, $j = 2$, $j' = 5$, and $k = 4$, i.e., we consider SNP-pair (X_3X_2), permutation Y_4, and the edge (X_2X_5) in Figure 5.1. Suppose that we already know that $O_{d_2}(X_3X_2, Y_4) = 2$, and event $D(X_3, Y_4) = \{S_{10}, S_{13}, S_{19}\}$. From Table 5.3, we have $(X_2X_5)_{\{0\to1\}\cup\{2\to1\}} = \{S_1, S_8, S_{10}\}$, and $(X_2X_5)_{\{1\to0\}\cup\{1\to2\}} = \{S_{13}\}$. Thus according to Theorem 5.5.1, we have $O_{d_2}(X_3X_5, Y_4) = O_{d_2}(X_3X_2, Y_4) + |\{S_{10}\}| - |\{S_{13}\}| = 2$. Note that by this way, we get the value of $O_{d_2}(X_3X_5, Y_4)$ from $O_{d_2}(X_3X_2, Y_4)$without scanning all individuals.*

So far, we have discussed the procedure to update the value of $O_{d_2}(X_iX'_j, Y_k)$ from $O_{d_2}(X_iX_j, Y_k)$ for a specific phenotype permutation Y_k. This procedure can be easily extended to handle all the permutations. From Theorem 5.5.1, for any permutation Y_k, to update the value of $O_{d_2}(X_iX'_j, Y_k)$ from $O_{d_2}(X_iX_j, Y_k)$, we need the value of $D(X_i, Y_k)$ and the genotype difference associated with edge $(X_jX'_j)$. Note that the genotype difference is fixed once the minimum spanning tree is built. Next, we discuss how to compute $D(X_i, Y_k)$ for all permutations $\{Y_1, Y_2, \cdots, Y_K\}$ in a batch mode in detail.

Let $D_K(X_i)$ be a list of M entries, with each entry corresponding to an individual. For each individual S_m, we record in $D_K(X_i)[m]$ the set of phenotypes satisfying $(X_i = 1 \wedge Y_k = 1)$. For example, consider the dataset in Table 5.1, we have that $D_K(X_3)[8] = \{Y_2, Y_3\}$. Table 5.4(a) shows the entries of $D_K(X_3)$. Only non-empty entries, i.e., $D_K(X_i)[m] \neq \emptyset$, are shown in the table. It is easy to see that, for any X_i and Y_k, we can get $D(X_i, Y_k)$ from $D_K(X_i)$ as follows: $D(X_i, Y_k)$ is the set of individuals whose corresponding entries in $D_K(X_i)$ contain Y_k as an element, i.e.,

$$D(X_i, Y_k) = \{S_m | Y_k \in D_K(X_i)[m]\}. \tag{5.1}$$

For example, using the example dataset in Table 5.1, from Table 5.4(a), we know that $D(X_3, Y_4) = \{S_{10}, S_{13}, S_{19}\}$.

For SNP-pair (X_iX_j), let $O_{d_2}(X_iX_j) = [O_{d_2}(X_iX_j, Y_1), O_{d_2}(X_iX_j, Y_2), \cdots, O_{d_2}(X_iX_j, Y_K)]$. From Theorem 5.5.1 and Equation (5.1), for any SNP-pair (X_iX_j) and an edge $(X_jX'_j) \in E(\mathcal{T})$, we can get $O_{d_2}(X_iX'_j)$ from $O_{d_2}(X_iX_j)$ using $D_K(X_i)$ and the genotype difference information associated with edge $(X_jX'_j)$. First, initialize $O_{d_2}(X_iX'_j) = O_{d_2}(X_iX_j)$. Next, for every m $(1 \leq m \leq M)$, if $Y_k \in D_K(X_i)[m]$, we update $O_{d_2}(X_iX'_j)$ as follows:

$$\begin{cases} \text{increase } O_{d_2}(X_jX'_j, Y_k) & \text{if } S_m \in (X_jX'_j)_{\{0\to1\}\cup\{2\to1\}}; \\ \text{decrease } O_{d_2}(X_jX'_j, Y_k) & \text{if } S_m \in (X_jX'_j)_{\{1\to0\}\cup\{1\to2\}}. \end{cases}$$

Example 5.5.3. *Following Example 5.5.2, we consider the two SNP-pairs (X_3X_2) and (X_3X_5), with (X_2X_5) being an edge of the tree in Figure 5.1. Assume that $D_K(X_3)$ is as shown in Table 5.4(a), and $O_{d_2}(X_3X_2) = [1, 1, 1, 2, 1]$. From Table 5.3, the genotype difference on edge (X_2X_5) is $(X_2X_5)_{\{0\to1\}\cup\{2\to1\}} = \{S_1, S_8, S_{10}\}$, and $(X_2X_5)_{\{1\to0\}\cup\{1\to2\}} = \{S_{13}\}$. For individual $S_m \in \{S_1, S_8, S_{10}\}$ ($S_m \in \{S_{13}\}$), we need to increase (decrease) the corresponding values in $O_{d_2}(X_3X_2)$ according to $D_K(X_3)$. Table 5.4(b) shows the updating process for $O_{d_2}(X_3X_5)$. Initially, $O_{d_2}(X_3X_5) = O_{d_2}(X_3X_2)$. For individual S_1, since its corresponding entry in $D_K(X3)$, $D_K(X3)[1] = \emptyset$, $O_{d_2}(X_3X_5)$ remains unchanged. For individual S_8,*

$D_K(X3)[8] = \{Y_2, Y_3\}$*, we increase the values of* $O_{d_2}(X_3X_5, Y_2)$ *and* $O_{d_2}(X_3X_5, Y_2)$ *by 1. Similarly, we increase and decrease the values in* $O_{d_2}(X_3X_5)$ *according to* $D_K(X3)$ *for the remaining individuals. The final result is* $O_{d_2}(X_3X_5) = [0, 2, 3, 2, 1]$.

Note that to get the value of $O_{d_2}(X_iX_j)$, using a brute force approach, we need to scan a $(2+K) \times M$ matrix consisting of the genotype of (X_iX_j) and permutations $\{Y_1, Y_2, \cdots, Y_K\}$ for the M individuals. In the previous example, to compute the value of $O_{d_2}(X_3X_5)$, the cost of the brute force approach is $(3 + 5) \times 24 = 192$. Using our approach, the total number of updates is $|D_K(X3)[8]| + |D_K(X3)[10]| + |D_K(X3)[13]| = 10$, which is significantly less than the cost of the brute force approach. More formally, given $D_K(X_i)$, the time complexity of updating $O_{d_2}(X_iX'_j)$ from $O_{d_2}(X_iX_j)$ is $O(w(X_jX'_j)K)$.

The procedure of updating $O_{d_2}(X_iX'_j)$ from $O_{d_2}(X_iX_j)$ can also be applied to update the remaining free variables $O_{d_3}(X_iX_j)$, $O_{f_2}(X_iX_j)$, $O_{f_3}(X_iX_j)$. Note that, to update $O_{f_2}(X_iX_j)$, $O_{f_3}(X_iX_j)$, we will need $F_K(X_i)$, which can be defined in a similar way to that of $D_K(X_i)$: for each individual S_m, we record in $F_K(X_i)[m]$ the set of phenotypes satisfying $(X_i = 2 \wedge Y_k = 1)$.

5.6 The TEAM Algorithm

TEAM examines SNP pairs through a double loop, where the outer loop visits a leaf node at a time, and the inner loop traverse the rest of the tree, starting from the parent node of the leaf. Let $O_{d_2d_3f_2f_3}(X_iX_j) = [O_{d_2}(X_iX_j), O_{d_3}(X_iX_j), O_{f_2}(X_iX_j), O_{f_3}(X_iX_j)]$. Let $L(\mathcal{T}) \in V(\mathcal{T})$ be the set of leaf nodes of the minimum spanning tree $\mathcal{T}$. For any *leaf* node $X_i \in L(\mathcal{T})$, let $AP(X_i) = \{(X_iX_j)|i \neq j, X_j \in V(\mathcal{T})\}$. Let X_a be the parent node of X_i. Since all SNPs are connected in $\mathcal{T}$, once we have $O_{d_2d_3f_2f_3}(X_iX_a)$, we can update all $O_{d_2}(X_iX_j) \in AP(X_i)$ by enumerating the edges in $E(\mathcal{T})$ in a breath-first traversal starting from X_a.

Example 5.6.1. *Consider the tree in Figure 5.1. Let* $X_i = X_3$ *and* $X_a = X_2$*. We have* $AP(X_3) = \{(X_3X_2), (X_3X_5), (X_3X_6), (X_3X_1), (X_3X_4)\}$*. Starting from* X_3*, a breadth first*

Algorithm 4: The TEAM Algorithm

Input: SNPs $X' = \{X_1, X_2, \cdots, X_N\}$, phenotype permutations $Y' = \{Y_1, Y_2, \cdots, Y_K\}$

Output: $\mathscr{T}(X_iX_j, Y_k)$ for all possible two-locus tests

```
compute and store all single-locus contingency tables;
build minimum spanning tree T;
for every X_i ∈ L(T), do
    compute D_K(X_i) and F_K(X_i);
    compute O_{d2d3f2f3}(X_iX_a);
    compute T(X_iX_a, Y_k) (1 ≤ k ≤ K) and output;
    EnumStack.push(O_{d2d3f2f3}(X_iX_a));
    while EnumStack ≠ ∅ do
        O_{d2d3f2f3}(X_iX_j) = EnumStack.pop();
        for every X'_j = adj(X_j) do
            update O_{d2d3f2f3}(X_iX'_j) from O_{d2d3f2f3}(X_iX_j);
            compute T(X_iX'_j, Y_k) (1 ≤ k ≤ K) and output;
            EnumStack.push(O_{d2d3f2f3}(X_iX'_j));
        end
    end
    delete X_i from T;
end
```

search will enumerate edges $\{(X_2X_5), (X_5X_6), (X_6X_1), (X_6X_4)\}$*, which can be utilized to update* $O_{d_2d_3f_2f_3}(X_iX_j)$ *for the SNP-pairs in* $AP(X_3)$.

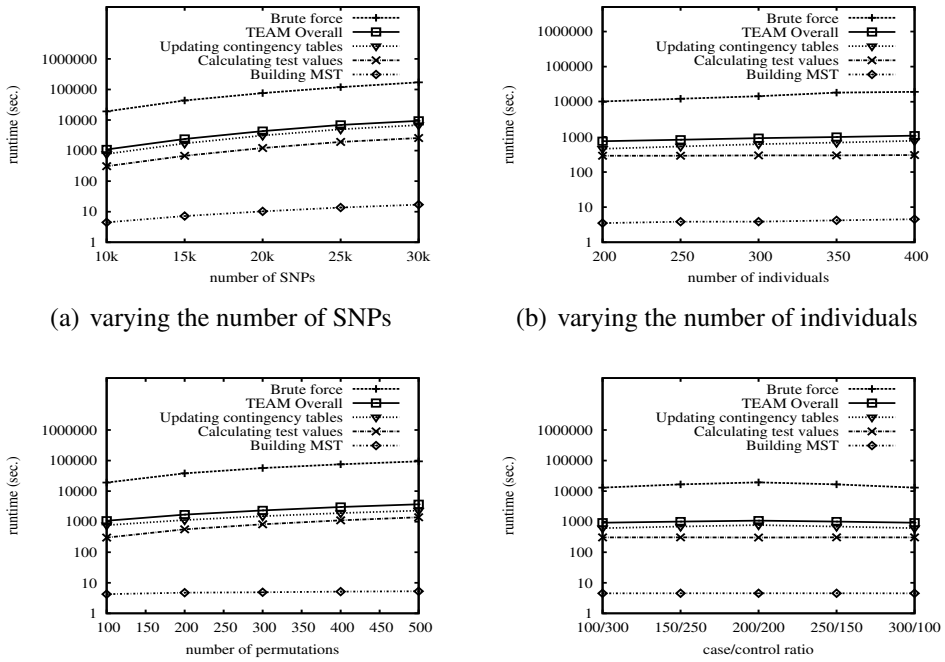
Once the SNP-pairs in $AP(X_i)$ have been processed, we delete X_i from $L(\mathcal{T})$, and repeat the same process for another leaf node. The overall algorithm is summarized in Algorithm 4. Given the SNPs $X' = \{X_1, X_2, \cdots, X_N\}$, phenotype permutations $Y' = \{Y_1, Y_2, \cdots, Y_K\}$, we first enumerate and store all single-locus contingency tables. We then build the minimum spanning tree $\mathcal{T}$, with genotype difference associated with each edge. For leaf node X_i, we compute $D_K(X_i)$, $F_K(X_i)$, and $O_{d_2d_3f_2f_3}(X_iX_a)$. This information is then used to incrementally update $O_{d_2d_3f_2f_3}(X_iX'_j)$ for all SNP-pairs in $AP(X_i)$. After processing $AP(X_i)$, we delete X_i from $\mathcal{T}$ and repeat the procedure for the remaining leaf nodes.

Time Complexity: The time complexity on generating all single-locus contingency tables and building the minimum spanning tree is $O(MNK)$ and $O(MN^2)$ respectively. The

time complexity to compute $D_K(X_i)$ and $F_K(X_i)$ for all SNPs is $O(MNK)$. The total updating cost for all $AP(X_i)$ is $O(W_{\mathcal{T}}NK)$. Thus the overall time complexity of TEAM is $O(MNK + MN^2 + W_{\mathcal{T}}NK)$. Note that the complexity of the brute force approach is $O(MN^2K)$. The number of SNPs N is the dominant factor.

Space Complexity: The dataset size is $O(M(N + K))$. The space needed to store all single-locus contingency tables is $O(NK)$. The size of tree $\mathcal{T}$ is $O(W_{\mathcal{T}})$. The size of $D_K(X_i)$ and $F_K(X_i)$ is $O(MK)$. Thus the total space complexity of TEAM is $O(M(N+K)+K(N+M)+W_{\mathcal{T}})$.

Note that we can do incremental computation using any exploration order. TEAM utilizes minimum spanning tree to update the contingency tables. The reason is that the cost of such update depends on the difference between the SNPs. The more similar they are, the lower the cost. Since minimum spanning tree has the minimum weight $W_{\mathcal{T}}$ over all spanning trees, using it to guide the computation leads to optimal efficiency. It is not absolutely necessary to use a minimum spanning tree. As long as the tree is close to a minimum spanning tree, we should expect good performance. An implementation issue in building the minimum spanning tree is that we need $O(N^2)$ space to store all pair-wise differences between the SNPs. In practice, we divide the SNPs into sub-groups of equal size. A minimum spanning tree is built for each group. Then the sub-trees are merged to a larger tree by randomly connecting leave nodes. The tree built in this way is an approximate minimum spanning tree. Our focus in this paper is not to build an optimal minimum spanning tree, but to use the tree structure for efficient updating. Please refer to (Eisner (1997); Graham and Hell (1985)) for surveys on minimum spanning tree construction. In the experiments, we show the performance evaluation using different spanning trees.



(a) varying the number of SNPs

(b) varying the number of individuals

(c) varying the number of permutations

(d) varying the case/contral ratio

Figure 5.2: Comparison between TEAM and the brute-force approach on human datasets under various experimental settings

5.7 Experimental Results

In this section, we present extensive experimental results on the performance of the TEAM algorithm. TEAM is implemented in C++. We first evaluate the efficiency of TEAM. Then we present the findings of epistasis detection in simulated human genome-wide study.

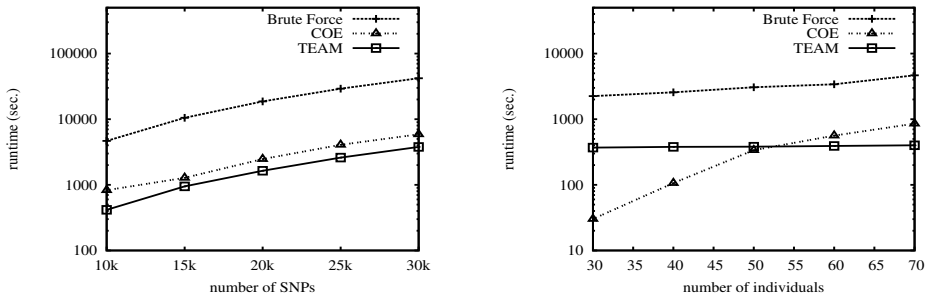
5.7.1 Efficiency Evaluation

We use both simulated human datasets and real mouse datasets for the efficiency evaluation experiments. The experiments are performed on a 2.6 GHz PC with 8G memory running Linux system.

Human data: The human datasets are generated by the simulator Hapsample (Wright et al. (2007)), which is publicly accessible from the website `http://www.hapsample.org`. We evaluate the performance of TEAM by comparing it with the brute force approach since there is no previous algorithm readily applicable to human datasets. Note that the brute-force

		TEAM		Updating by Random Tree		Updating by Linear Tree	
	Settings	**Tree weight**	**Pruning ratio**	**Tree weight**	**Pruning ratio**	**Tree weight**	**Pruning ratio**
# SNPs	10k	17.721%	94.104%	53.326%	88.722%	53.158%	89.210%
	20k	18.692%	93.981%	52.881%	88.895%	52.851%	89.390%
	30k	19.314%	93.802%	53.011%	88.823%	52.946%	89.380%
# Individuals	200	16.641%	94.376%	53.358%	88.749%	53.179%	89.205%
	300	17.342%	94.209%	53.343%	88.730%	53.142%	89.213%
	400	17.721%	94.104%	53.326%	88.722%	53.158%	89.210%
# Permutations	100	17.721%	94.104%	53.326%	88.722%	53.158%	89.210%
	300	17.721%	94.105%	53.326%	88.724%	53.158%	89.212%
	500	17.721%	94.104%	53.326%	88.724%	53.158%	89.212%
Case/control ratio	100/300	17.721%	97.049%	53.326%	94.355%	53.158%	94.599%
	200/200	17.721%	94.104%	53.326%	88.722%	53.158%	89.210%
	300/100	17.721%	97.049%	53.326%	94.355%	53.158%	94.599%

Table 5.5: The tree weight and the proportion of the individuals pruned by TEAM on the human datasets



(a) varying the number of SNPs (b) varying the number of individuals

Figure 5.3: Comparison between TEAM, COE, and the brute force approach on mouse datasets under various experimental settings

approach is very time consuming, we use a moderate number of SNPs and permutations in the experiments so that the brute-force approach can finish in a reasonable amount of time. Unless otherwise specified, the default experimental setting is the following: #individuals = 400, #SNPs=10,000, #permutations=100, and the case/control ratio is 1. These experimental settings are chosen to demonstrate the efficiency gain offered by TEAM over the brute-force implementation. TEAM can handle much larger datasets. The performance of TEAM is expected to follow the same trends presented in this section.

TEAM contains three major components: building the minimum spanning tree, updating the contingency tables, and calculating the actual test values. Note that TEAM can be applied to any statistics defined on the contingency table. With different statistics, the only difference in runtime would be caused by the last component calculating the statistics. In the

experiments, we choose chi-square test as our statistic. Figure 5.2 shows the running time comparison of TEAM and the brute-force approach using different experimental settings. The y-axis is in logarithm scale. In these figures, we also show the detailed runtime of these three components.

Table 5.5 shows the percentage of individuals pruned by TEAM under different experimental settings. Since in theory we can update the contingency tables in any exploration order, in the table, we also show the pruning effect of using a random spanning tree and a linear spanning tree to guide the updating process. The random spanning tree is generated by starting from a randomly picked SNP and growing edges that connect the remaining SNPs in a random order. The linear tree is a single path connecting all SNPs sequentially. From the table, we can see that TEAM prunes more effectively than the other two updating methods. In the table, we also show the ratio of the tree weights and the size of the SNP dataset, i.e., $W_{\mathcal{T}}/(M \times N)$, which is a determining factor of the pruning ratio. Note that varying the number of permutations and the case/control ratio does not effect the tree being built.

Figures 5.2(a) depicts the runtime comparison when varying the number of SNPs. TEAM is more than an order of magnitude faster than the brute-force approach. Among the three components of TEAM, the procedures on building the minimum spanning tree and calculating test values only take a small portion of the total runtime of TEAM. The runtime of TEAM is dominated by the cost of updating the contingency tables. As will be shown later, TEAM prunes most of the individuals when updating the contingency tables. In Figures 5.2(b), 5.2(c), and 5.2(d), we can also observe a similar one to two orders of magnitude speedup of TEAM over the brute force approach when varying the number of individuals, the number of permutations, and the case/control ratio.

Mouse data: The mouse datasets is extracted from a set of combined SNPs from the 10k GNF mouse dataset and the 140k Broad/MIT mouse dataset. This merged dataset has 156,525 SNPs for 71 mouse strains. The missing values in the dataset are imputed using NPUTE (Roberts et al. (2007)). We compare TEAM and COE algorithm, which is specifi-

Dataset	Significant SNP-Pair	Chromosome and Location	FDR	FWER
1	(rs768529, rs3804940)*	(chr1: 51946762, chr3: 7520545)	0.00067	0
	(rs768529, rs756084)	(chr1: 51946762, chr3: 7536149)	0.00067	0
	(rs768529, rs779742)	(chr1: 51946762, chr3: 7558058)	0.00067	0
	(rs768529, rs1872393)	(chr1: 51946762, chr3: 7546236)	0.00067	0.004
	(rs768529, rs779744)	(chr1: 51946762, chr3: 7555121)	0.00067	0.004
	(rs768529, rs6764561)	(chr1: 51946762, chr3: 7514592)	0.00067	0.004
2	(rs10495728, rs521882)*	(chr2: 22811773, chr8: 16688797)	0.004	0.004
3	(rs1016836, rs2783130)*	(chr10: 31935845, chr13: 79068161)	0	0
4	(rs648519, rs1012273)*	(chr11: 98972936, chr16: 58525067)	0.002	0.002

Table 5.6: Identified significant SNP-pairs in the simulated human GWAS datasets

cally designed for association study in mouse datasets. The default experimental setting is as follows: #individuals = 70, #SNPs=10,000, #permutations=100, and the case/control ratio is 1.

Figure 5.3 shows the comparison results. In the figure, we also plot the runtime of the brute force approach. Figure 5.3(a) shows the runtime of the three approaches when varying the number of SNPs. It is clear that both TEAM and COE are orders of magnitude faster than the brute force approach. TEAM is about twice faster than COE. Figure 5.3(b) shows the runtime comparison when varying the number of individuals. From the figure, COE is more suitable for datasets having small number of individual. As the number of individuals increases, the TEAM algorithm becomes more efficient than COE. Note that in human study, the number of individuals usually ranges up to thousands, much larger than that in typical mouse datasets.

5.7.2 Epistasis Detection in Simulated Human GWAS

In this section, we report the results of epistasis detection using simulated human GWAS data generated by Hapsample. In total, we generate 4 datasets, each of which has 112,036 SNPs for 250 cases and 250 controls. In each dataset, a disease causal interacting SNP-pair is embedded. The embedded SNP-pairs are: (rs768529, rs3804940) in dataset 1, (rs10495728, rs521882) in dataset 2, (rs1016836, rs2783130) in dataset 3, and (rs648519, rs1012273) in

dataset 4. We use standard chi-square test with 500 permutations. Similar results can be found by using likelihood-ratio test.

With an overall FDR threshold of 0.005, Table 5.6 shows the identified significant SNP-pairs using TEAM. TEAM successfully identified the embedded SNP-pairs in all simulated datasets. The embedded SNP-pairs are labelled with stars "*". The table shows the SNP loci on the genome. For example, in dataset 1, we embed SNP-pair rs768529 and rs3804940, which are located on chromosome 1 at position 51946762 base-pair and chromosome 3 at 7520545 base-pair respectively. The FWER for each reported SNP-pair is also shown. Note that, for a SNP-pair, a FDR (or FWER) value of 0 indicates that permutation tests do not generate any test value larger than value of the reported SNP-pair. In dataset 1, except for the embedded SNP-pair (rs768529, rs3804940), 5 other SNP-pairs are also reported. One of the embedded SNP, rs768529, is involved in all the 5 pairs. A closer look at the other SNPs in the reported SNP-pairs shows that they are all adjacent to the embedded SNP rs3804940. The normalized linkage disequilibrium (Lewontin and Kojima (1960)) between rs3804940 and the other 5 SNPs are D'(rs3804940, rs756084)= 1, D'(rs3804940, rs779742)= 0.477, D'(rs3804940, rs1872393)= 0.442, D'(rs3804940, rs779744)= 0.442, and D'(rs3804940, rs6764561)= 0.454, indicating there is strong linkage disequilibrium between them.

5.8 Conclusion

The large number of SNPs genotyped in the genome-wide scale poses great computational challenges in two-locus epistasis detection. The permutation test used for proper error rate controlling makes the problem computationally even more intensive. In this chapter, we propose an efficient algorithm, TEAM, for epistasis detection human GWAS. TEAM has the same strength as the recently developed epistasis detection methods, i.e., it guarantees to find the optimal solution. Compared to existing methods, TEAM is more efficient in large sample study, and offers broader applicability. Existing methods designed for homozygous SNPs

cannot be used for human data where SNPs are commonly heterozygous. TEAM, on the other hand, can handle both homozygous and heterozygous SNPs. Since it exhaustively enumerate all SNP-pairs, TEAM can be used to control the FWER and the FDR, both of which are widely used in controlling error in GWAS; while previous methods only control the FWER. Existing methods need to exam the formulation of the statistic. TEAM is focused on efficiently updating contingency tables rather than any specific statistic. It can therefore be use for any statistical test based on contingency table regardless of its formulation.

Chapter 6

Discussion

Driven by the advancement of cost-effective and high-throughput genotyping technologies, genome-wide association studies (GWAS) have revolutionized the field of genetics by providing new ways to identify genetic factors that influence phenotypic traits. Most of these studies have used a single-locus analysis strategy, in which each genetic variants is tested individually for association with a phenotype. However, many common diseases are complex traits and caused by interactions between loci. The identification of *epistasis*, or gene-gene interaction, is thus much preferable over a single-locus approach. Complete genome-wide epistasis detection has previously been considered *intractable* due to several thorny problems in both statistical and computational aspects. The statistical challenges are to develop effective tests to capture gene-gene interactions, and to properly control error rates due to the large number of correlated tests. The computational challenge is the intensive computation burden of searching for interactions between millions of variants spread across the entire genome. These challenges interact with each other and must be handled together.

This thesis presents several algorithms that enable efficient and exhaustive epistasis detection in the whole-genome. A summary of these methods can be found in Table 6.1. The first is the FastANOVA algorithm. It incorporates a large permutation test for family-wise error rate (FWER) controlling. By indexing the genetic variants and utilizing an upper bound on the test statistic, FastANOVA dramatically prunes the search space and reduces redun-

Algorithm	Trait	Genotype	Error Type	Sample Size	Supported Test
FastANOVA	quantitative	binary	FWER	less than a hundred	ANOVA test
FastChi	binary	binary	FWER	less than a hundred	chi-square test
COE	binary	binary	FWER	less than a hundred	convex test
TEAM	binary	any	FWER & FDR	hundreds to thousands	test based on contingency tables

Table 6.1: Algorithms and their corresponding problem settings for epistasis detection in genome-wide association study

dant computation. Consequently, FastANOVA only needs to examine a very small portion of the variants without the risk of missing any significant interaction. The principle used in FastANOVA can also be applied to the chi-square test, which is widely used in case-control studies. Based on the observation that many commonly used statistics are convex functions, a unified algorithm COE is developed, which can be applied to all convex statistics. A more general algorithm TEAM is also presented. TEAM is applicable to any statistical test that is based on a contingency table. It is suitable for large sample human studies and supports the control of FWER and false discovery rate (FDR) , both of which are effective methods for controlling error rates in GWAS. Extensive experimental results demonstrate the efficiency of the proposed algorithms.

Bibliography

Balding, D. J. (2006). A tutorial on statistical methods for population association studies. *Nature Reviews Genetics*, 7(10):781–791. 10, 57

Bohringer, S., Hardt, C., Miterski, B., Steland, A., and Epplen, J. T. (2003). Multilocus statistics to uncover epistasis and heterogeneity in complex diseases: revisiting a set of multiple sclerosis data. *European Journal of Human Genetics*, 11:573–584. 57

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press. 58, 63, 72, 73

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Monterey, Calif., U.S.A.: Wadsworth, Inc. 10

Carlborg, O., Andersson, L., and Kinghorn, B. (2000). The use of a genetic algorithm for simultaneous mapping of multiple interacting quantitative trait loci. *Genetics*, 155(4):2003–2010. 11

Carlson, C. S., Eberle, M. A., Kruglyak, L., and Nickerson, D. A. (2004). Mapping complex disease loci in whole-genome association studies. *Nature*, 429:446–452. 4

Chi, P., Duggal, P., Kao, W., and et al. (2006). Comparison of snp tagging methods using empirical data: association study of 713 snps on chromosome 12q14.3-12q24.21 for asthma and total serum ige in an african caribbean population. *Genetic Epidemiology*, 30(7):609–619. 11

Churchill, G. A. and Doerge, R. W. (1994). Empirical threshold values for quantitative trait mapping. *Genetics*, 138(3):963–971. 5

Cordell, H. J. (2009). Detecting genegene interactions that underlie human diseases. *Nature Reviews Genetics*, 10:392–404. 1, 4

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press and McGraw-Hill. 75, 82

Curtis, D., North, B. V., and Sham, P. C. (2001). Use of an artificial neural network to detect association between a disease and multiple marker genotypes. *Annals of Human Genetics*, 65:95–107. 10

Doerge, R. W. (2002). Multifactorial genetics: Mapping and analysis of quantitative trait loci in experimental populations. *Nature Reviews Genetics*, 3:43–52. 10

Dong, C. and et al. (2008). Exploration of genegene interaction effects using entropy-based methods. *European Journal of Human Genetics*, 16:229–235. 57, 61

Dudoit, S. and van der Laan, M. J. (2008). *Multiple testing procedures with applications to genomics*. Springer. 74, 76

Eisner, J. (1997). State-of-the-art algorithms for minimum spanning trees: A tutorial discussion. *Manuscript,University of Pennsylvania*. 88

Evans, D. M., Marchini, J., Morris, A. P., and Cardon, L. R. (2006). Two-stage two-locus models in genome-wide association. *PLoS Genetics*, 2: e157. 11

Graham, R. L. and Hell, P. (1985). On the history of the minimum spanning tree problem. *Ann. History Comput.*, 7:43–57. 88

Halperin, E., Kimmel, G., and Shamir, R. (2005). Tag snp selection in genotype data for maximizing snp prediction accuracy. In *Proc. ISMB*. 11

Hoh, J. and Ott, J. (2003). Mathematical multi-locus approaches to localizing complex human trait genes. *Nature Reviews Genetics*, 4:701–709. 10

Hoh, J., Wille, A., Zee, R., Cheng, S., Reynolds, R., Lindpaintner, K., and Ott, J. (2000). Selecting snps in two-stage analysis of disease association data: a model-free approach. *Annals of Human Genetics*, 64:413–417. 11

Ideraabdullah, F., Casa-Espern, E., and et al. (2004). Genetic and haplotype diversity among wild-derived mouse inbred strains. *Genome Research*, 14(10a):1880–1887. 1

Lewontin, R. C. and Kojima, K. (1960). The evolutionary dynamics of complex polymorphisms. *Evolution*, 14(4):458–472. 93

Liu, H. and Motoda, H. (1998). *Feature selection for knowledge discovery and data mining*. Boston: Kluwer Academic Publishers. 11

Mielke, P. W. and Berry, K. J. (2001). *Permutation Methods: A Distance Function Approach*. Springer. 14

Miller, R. G. (1981). *Simultaneous Statistical Inference*. Springer Verlag New York. 4

Moore, J. H., Gilbert, J. C., Tsai, C.-T., F-T. Chiang, T. H., Barney, N., and White, B. C. (2006). A flexible computational framework for detecting, characterizing, and interpreting statistical patterns of epistasis in genetic studies of human disease susceptibility. *Journal of Theoretical Biology*, 241(2):252–261. 11

Musani, S., Shriner, D., Liu, N., Feng, R., Coffey, C., Yi, N., Tiwari, H., and Allison, D. (2007). Detection of gene x gene interactions in genome-wide association studies of human population data. *Human Heredity*, 63(2):67–84. 1

Nakamichi, R., Ukai, Y., and Kishino, H. (2001). Detection of closely linked multiple quantitative trait loci using a genetic algorithm. *Genetics*, 158(1):463–475. 11

Nelson, M. R., Kardia, S. L., Ferrell, R. E., and Sing, C. F. (2001). A combinatorial partitioning method to identify multilocus genotypic partitions that predict quantitative trait variation. *Genome Research*, 11:458–470. 11

Pagano, M. and Gauvreau, K. (2000). *Principles of Biostatistics*. Pacific Grove, CA: Duxbury Press. 2, 9, 14, 57, 59

Pesarin, F. (2001). *Multivariate Permutation Tests*. Wiley. 14

Province, M. A., Shannon, W. D., and Rao, D. C. (2001). Classification methods for confronting heterogeneity. *Advances in Genetics*, 42:273–286. 10

Ritchie, M. D., Hahn, L. W., Roodi, N., Bailey, L. R., Dupont, W. D., Parl, F. F., and Moore, J. H. (2001). Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. *American Journal of Human Genetics*, 69:138–147. 11

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. In *Proc. ISMB*. 26, 51, 69, 91

Saxena, R., Voight, B. F., Lyssenko, V., Burtt, N. P., de Bakker, P. I. W., Chen, H., Roix, J. J., Kathiresan, S., Hirschhorn, J. N., Daly, M. J., Hughes, T. E., Groop, L., Altshuler, D., Almgren, P., Florez, J. C., Meyer, J., Ardlie, K., Bengtsson Bostrm, K., Isomaa, B., Lettre, G., Lindblad, U., Lyon, H. N., Melander, O., Newton-Cheh, C., Nilsson, P., Orho-Melander, M., Rstam, L., Speliotes, E. K., Taskinen, M.-R., Tuomi, T., Guiducci, C., Berglund, A., Carlson, J., Gianniny, L., Hackett, R., Hall, L., Holmkvist, J., Laurila, E., Sjgren, M., Sterner, M., Surti, A., Svensson, M., Svensson, M., Tewhey, R., Blumenstiel, B., Parkin, M., DeFelice, M., Barry, R., Brodeur, W., Camarata, J., Chia, N., Fava, M., Gibbons, J., Handsaker, B., Healy, C., Nguyen, K., Gates, C., Sougnez, C., Gage, D., Nizzari, M., Gabriel, S. B., Chirn, G.-W., Ma, Q., Parikh, H., Richardson, D., Ricke, D., and Purcell, S. (2007). Genome-Wide Association Analysis Identifies Loci for Type 2 Diabetes and Triglyceride Levels. *Science*, 316(5829):1331–1336. 1, 5

Scuteri, A., Sanna, S., Chen, W.-M., and et al. (2007). Genome-wide association scan shows genetic variants in the fto gene are associated with obesity-related traits. *PLoS Genetics*, 3(7):1200–1210. 5

Sebastiani, P., Lazarus, R., Weiss, S. T., Kunkel, L. M., Kohane, I. S., and Ramoni, M. F. (2003). Minimal haplotype tagging. *PNAS*, 100(17):9900–9905. 11

Segr, D., DeLuna, A., Church, G. M., and Kishony, R. (2005). Modular epistasis in yeast metabolism. *Nature Genetics*, 37:77–83. 4

Sherriff, A. and Ott, J. (2001). Applications of neural networks for gene finding. *Advances in Genetics*, 42:287–297. 10

The Wellcome Trust Case Control Consortium (2007). Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. *Nature*, 447:661–678. 1

Thomas, D. C. (2004). *Statistical methods in genetic epidemiology*. Oxford Univeristy Press, Oxford. 57

Weedon, M., Lettre, G., Freathy, R., and et al. (2007). A common variant of hmga2 is associated with adult and childhood height in the general population. *Nature Genetics*, 39:1245–1250. 5

Westfall, P. H. and Young, S. S. (1993). *Resampling-based Multiple Testing*. Wiley, New York. 74, 76

Wright, F. A., Huang, H., Guan, X., Gamiel, K., and et al. (2007). Simulating association studies: a data-based resampling method for candidate regions or whole genome scans. *Bioinformatics*, 23(19):2581–2588. 89

Zhang, H. and Bonney, G. (2000). Use of classification trees for association studies. *Genetic Epidemiology*, 19:323–332. 10

Zhang, X., Pan, F., Xie, Y., Zou, F., and Wang, W. (2010). COE: a general approach for efficient genome-wide two-locus epistatic test in disease association study. *Journal of Computational Biology*, 17(3):401–415. 7

Zhang, X., Zou, F., and Wang, W. (2008). FastANOVA: an efficient algorithm for genome-wide association study. In *Proc. KDD*. 6, 59

Zhang, X., Zou, F., and Wang, W. (2009). FastChi: an efficient algorithm for analyzing gene-gene interactions. In *Proc. PSB*. 7, 35, 59, 69

Zhao, J., Boerwinkle, E., and Xiong, M. (2005). An entropy-based statistic for genomewide association studies. *Am J Hum Genet*, 77:27–40. 57, 61