이학박사학위논문

Sparse hypergraph clustering and its applications

희박한 하이퍼그래프의 클러스터링과 그 활용

2017년 8월

서울대학교 대학원
수학교육과
김 선 아

Sparse hypergraph clustering and its applications

A dissertation
submitted in partial fulfillment
of the requirement for the degree of
Doctor of Philosophy

by

Sun Ah Kim

Dissertation Director : Professor Yun Joo Yoo

Department of Mathematics Education
Seoul National University

August 2017

Abstract

Clustering is one of the most popular methods to extract meaningful patterns from data. In genomics, increasingly large amounts of DNA sequencing data are being generated. Developing effective clustering tools appropriate for each data structure is a major challenge. In this thesis, we develop the CLQ and CLQ-D clustering algorithm to partition SNP sequence data using theoretical graph-based approaches. Based on these clustering algorithms, the LD block construction method, Big-LD, is able to detect the LD blocks of SNP sequence data using interval graph modeling of the clustering results. A sparse graph is defined as a graph in which the actual number of edges is much less than the possible number of edges. Real-world data including biological, social, and internet network data can be modeled as sparse graphs. Due to the structural characteristic of SNP data, graph models constructed for the clustering algorithm and LD block construction algorithm have a sparse structure, which facilitates the efficient operation of the algorithm in terms of time and memory usage. The Big-LD algorithm detects LD blocks including "holes", which are not allowed in the previous methods. Based on the LD block structure constructed by the Big-LD algorithm, we investigated the relationships between big LD block structure and biological phenomena using the HapMap phase 3 data and phase 1 data of the 1000 Genomes Project. The LD block boundaries detected by the Big-LD algorithm coincided better with the recombination hotspots than previous methods. In addition, we demonstrate that the comparison of LD block structures can provide additional information about positive selection using the results applied to the candidate regions suggested by previous research. By generalizing the Big-LD algorithm, which is designed to partition SNP sequence data

into blocks, we suggest four clustering algorithms—PSHSC, PSHRC, PSHSQ, and PSHRQ—for the sparse hypergraph partitioning problem. Simulation experiments demonstrated that the algorithms generate high-quality partitions in terms of global and local quality measures. The partitioning results closely agreed with the true underlying cluster structures of simulated hypergraphs. We also applied the developed algorithm to the problem of predicting protein complexes in yeast protein-protein interaction network data, and confirm its potential as a tool for clustering biological network datasets.

Keywords : hypergraph, clustering, maximum weight independent set, clique, linkage disequilibrium, LD block, single nucleotide polymorphism, positive selection

Student Number : 2009-21501

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Clustering is one of the most popular methods to extract meaningful patterns from data. The clustering task for a set of data objects involves identifying groups of objects sharing similar features in the same cluster compared to the objects in different clusters. Clustering may allow overlap between clusters. However, many clustering problems impose a goal of dividing the data objects into disjoint nonempty sets. This type of clustering is called "partitioning". In partitioning, the data points are assigned into only one cluster (*partition*) based on their similarity and dissimilarity. The outcomes of clustering can reveal the underlying structure of the data and lead to new scientific discoveries.

While clustering can be applied to the data with various contexts and types, there is no gold standard to deal with the problem in a universal way. The clustering problem can be defined differently in different situations. Sometimes, an optimal solution cannot be attained due to the impracticality of the application of the theory. Therefore, it is important to adopt appropriate data modeling for specific problems and to apply tailored clustering methods to the data in light of the objectives of problem solving.

In genomics, ever-larger quantities of DNA sequencing data are being generated, and the discovery of structural characteristics and drawing of biological interpretations from the genomic data has become a major challenge. Single nucleotide

polymorphisms (SNP) are sequence variations in genomes. SNPs are analyzed to gain insight into gene-trait associations as well as population genetics [GBH+03, Sla08, TMP+03, SVF+07, GSK+10, WAZ+02, SMB+07, LH02, JKN01, Man10, WLZ15]. Clustering has several applications in SNP-based genotype data analysis. It is useful in the identification of tag SNPs that represent groups of SNPs in strong linkage disequilibrium (LD) with previously studied tag SNPs. LD is the non-random association of the alleles of genetic markers. Selection of a set of tag SNPs requires the identification of clusters of SNPs with strong LD. Once these clusters are identified, the best tag SNP for a cluster is chosen to represent the other SNPs in the cluster. The tag SNPs can capture the effects of untyped (data unavailable) causal SNPs through an indirect association, which allows the strategic planning of genotyping and saving of resources. In a genetic association study, an array comprising only tag SNPs that are tailored for the study can be used for genotyping, with the reduced cost guaranteeing a certain degree of power to capture the effect of causal SNPs that are not included in the array [Str04, ZQL+04, HKS05, ENK+07, ZRM+05].

Another recognized clustering problem based on genomic data is the identification of LD blocks in the genome. An LD block refers to a group of consecutive genetic markers in strong LD with each other. LD block construction can be seen as a block partitioning problem, which requires an additional constraint to be added to the clustering algorithm so that the clustered SNPs are consecutively positioned. In genome sequence data, the distribution of LD has been theorized as being reciprocally proportional to the distance between markers [RCB+01a]. However, actual LD patterns of SNP data are more complicated. Mosaic patterns are evident, with SNPs in weak LD with other SNPs being located in the middle of strongly correlated SNPs. The disruption caused by low LD SNPs in the high LD region are

referred to as "holes" [WP03]. The conventional methods of LD block clustering avoid including holes in the block. This approach seemed reasonable at the time of development since the methods have been developed using less dense genomic data in the years before next generation sequencing technology made it possible to obtain whole-genome sequencing data. However, the LD patterns of dense sequencing data with the genotyping of every single-base marker has revealed that the mosaic patterns of LD are much more complex than had been thought. The complexity means that excluding holes in LD blocks will produce blocks that are too short, rather than revealing the structure of LD formations extended over a wide range in the genome. In particular, the extended LD around a genetic variant is considered evidence of positive selection [SRH$^+$02, VKWP06, SVF$^+$07]. Therefore, obtaining large LD structures from whole-genome sequencing data could provide support for the population genetic hypothesis.

Clustering problems have been tackled using various types of models including modeling based on distance metrics like the K-means algorithm, statistical modeling such as principal component analysis, artificial neural network, and graph modeling [HW79, WEG87, BBC02, CPMC96, KWR$^+$01, Sch07, SM00, HS00, ZCY09, FTT04]. Graph-based approaches allow detailed structural modeling of data objects, which makes it possible to devise a clustering algorithm tailored for the specific situation. In graph modeling, a set of objects and their relatedness can be described by a graph with vertices representing the objects and edges representing the pairwise relationship between the objects (Figure 1.1(A)). For example, SNP sequence genotype data can be modeled using a graph whose vertices represent SNPs, with the two vertices joined by edges with weights assigned according to the values of pairwise LD measures, such as D' or r^2, obtained from the genotype data [Lew64]. A group of vertices of a graph model of SNP genotype data can

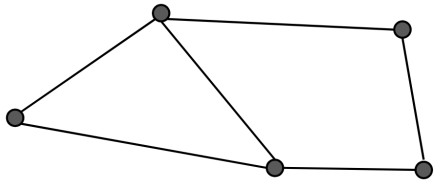
be taken as a cluster if every vertex in the subset is joined by edges with bigger weights. However, the edges linking the vertices of the group and the vertices outside the group have a relatively small weight. These clusters of SNPs can be used to select for tag SNPs. A sparse graph is defined as a graph in which the number of edges is much less than the number of possible edges. In many areas, such as social interaction analysis, biology, internet network analysis, and finance, data are often represented by sparse graphs [SXZF08]. SNP genotype data also can be depicted by a sparse graph representing the pairwise LD structure in SNPs. Since the strength of LD tends to diminish as the distance between two SNPs increases, most SNPs are in high LD with only a small proportion of SNPs in a chromosome. Using this property, a sparse graph can be constructed if the edges represent high pairwise LD between a pair of nodes representing SNPs; this helps in reducing the time and memorized information required to analyze the graph. Another application of sparse graph modeling is the protein-protein interaction (PPI) network. Usually, proteins interact with each other to regulate the biological functions, rather than act as isolated entities; hence, analysis of modules or complexes of proteins based on protein-protein interactions (PPI) facilitates deeper understanding of the biological process and mechanism in molecular systems [JZL$^+$14]. PPI can be presented in a graph (PPI network), where nodes represent proteins and two nodes are joined by an edge if the corresponding two proteins interact with each other. By analyzing the structural and topological properties of PPI networks and clustering of densely connected proteins in the PPI network, functional complexes can be detected.

In SNP sequencing data, the clusters of SNPs chosen by their LD relationship may not be physically consecutive. SNPs that are not included in a cluster but which are located in between SNPs in the cluster may be also considered as candidates for cluster membership if we wish to find big LD blocks that allow for holes.

Following this specific requirement, the SNP cluster can be regarded as an interval that ranges from a starting SNP to an ending SNP of the cluster. If we consider these intervals corresponding to LD-based clusters as a new level of objects, then another graph model at a second level can be constructed. In this *interval graph*, the intervals are represented by vertices. Overlap between intervals represents the incidence between two vertices (Figure 1.1(B)). This interval graph modeling approach provides a multi-level strategy to identify hidden structures from the data. LD block partitioning can be conducted using the second-level interval graph structure, which makes the block partitioning more natural, since the overlapping intervals (vertices joined by an edge in the interval graph) should be a block of consecutive SNPs. At each construction level, the graph model can maintain sparse structural properties because of the LD structure and the sequential characteristic of the SNP data. Thus, even a multi-level algorithm can run in real time.

The membership of SNPs in an LD-based cluster can be also depicted by a *hypergraph*, which is generalization of a graph by defining the edges (called *hyperedges*) consisting of more than two vertices (Figure 1.1(C)). Hyperedges can represent the first-level community membership of the objects. The overlap of membership between different hyperedges can be also depicted by a graph in second-level modeling, which is actually a dualization by a *line graph* of the hypergraph. By clustering the line graph of the hypergraph, we can solve a clustering problem given with the community membership information. Multi-level clustering algorithms might operate effectively, especially for a clustering problem that can be converted into a sparse hypergraph. PPI networks are scale-free networks whose degree distributions follow a power-law [JZL$^+$14]. Therefore, we can efficiently search locally dense sub-networks (i.e. maximal cliques) of a PPI network, since the number of sub-networks linearly increases with the number of proteins. With the detected

(A) Graph



(B) Interval representation and its corresponding interval graph

(C) Hypergraph

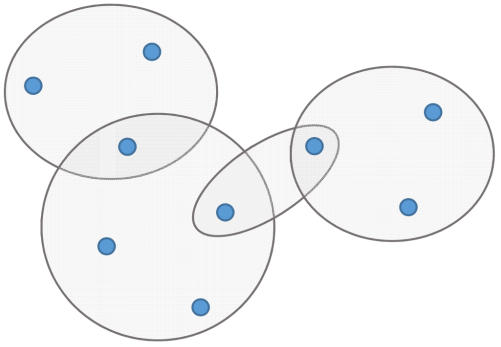


Figure 1.1: Examples of a graph (A), an interval graph (B), and a hypergraph (C)

sub-networks of a PPI network, we can construct a hypergraph using hyperedges representing the subnetworks and apply a multi-level clustering algorithm for the constructed hypergraph.

For clustering problems, there are two categories of methods. One is a sequence of methods that finds the best clusters when the number of clusters is fixed to a certain number. The other category of methods determines the number of clusters and also identifies the clusters. The latter has more complexity by introducing another variable, the number of clusters (usually denoted by k) in the optimization. For the clustering problems of SNP genotype data or PPI networks presented above, the number of clusters cannot be predetermined since there is no biological foundation to assume a fixed number for the clusters of SNPs or protein complexes. Therefore, we need a method to determine the number of clusters for DNA sequencing data of SNPs or protein complexes.

1.2 Objectives and contributions of the thesis

The main objectives and contributions in this work are summarized as follows:

1. We aim to develop a clustering algorithm for SNP genotype data based on LD structure. We propose two clustering algorithms, CLQ and CLQ-D, to partition SNPs into clusters such that every two SNPs in a cluster are in strong LD using the notion of clique.

2. We aim to develop an LD block construction algorithm that allows "holes" in the block. We introduce a new LD block construction algorithm, Big-LD, based on multi-level graph modeling using the notion of the interval graph. In Big-LD, SNP clusters detected by the CLQ-D algorithm represented using an interval graph model and LD block partitions are constructed by applying an algorithm to determ-

ine the number of clusters using the concept of Maximum Weight Independent Set (MWIS). The data demonstrate that the Big-LD algorithm produces a more optimized solution in the clustering compared to previous methods. Furthermore, the LD block boundaries determined using the Big-LD algorithm better agree with the recombination hotspots (biologically expected regions of low LD) than results obtained by previous LD block construction methods. The Big-LD algorithm produces more invariant results for the changes in the marker density of genomic data, which can be seen as the missing data situation, compared to previous methods.

3. We aim to reveal the relationships between the big LD block structure detected by the Big-LD algorithm and population genetics phenomena, especially positive selection. By comparing the LD blocks obtained by the Big-LD algorithm to the previously reported candidate regions for positive selection, we show that the big LD block structure might be used to find positive selection loci. Moreover, based on the analysis of the big LD block structure, we provide additional evidence for positive selection in the three gene regions (SLC30A9, PDE11A, and BCAS3) for East Asian and European populations.

4. We aim to generalize the multi-level clustering approach of the Big-LD algorithm to the clustering problem by using hypergraph modeling and the dualization by line graph. The strategy to determine the number of clusters using MWIS for sparse hypergraph models with a low number of edges. We show that the clustering results obtained by the suggested hypergraph clustering algorithm show high-quality partitioning results in terms of global and local quality measures, and cluster validation measures compared to previous methods. Additionally, we apply this clustering algorithm to predict protein complexes in the yeast PPI network data, and compare the performance between the hypergraph clustering algorithm and the existing methods.

1.3 Organization of the thesis

In chapter 2, we define various types of graphs, such as edge or vertex weighted graph, interval graph, intersection graph, line graph, and hypergraph, and detail important notions related to these graphs.

In chapter 3, we introduce the details of SNP clustering algorithms, CLQ and CLQ-D, and the LD block construction algorithm, Big-LD. Using data from the 1000 Genomes Project and the HapMap data, we compare our algorithm to other LD block construction algorithms based on statistical or sequence pattern analysis regarding the characteristic of the LD block structure and population genetics phenomena, such as recombination hotspots.

In chapter 4, we analyze the LD block structures obtained using the Big-LD algorithm for several previously suggested candidate regions for positive selection, and compare the results to other methods detecting positive selection loci. We show the potential value of the Big-LD algorithm as a detection tool for positive selection, and also show that LD block comparisons can provide additional information for positive selection in regions of SLC30A19, PDE11A, and BCAS3 for East Asian and European populations based on the LD block patterns.

In chapter 5, we propose partitioning algorithms for sparse hypergraph as a generalization of the Big-LD algorithm. We suggest four algorithms (PSHSC, PSHRC, PSHSQ, and PSHRQ), which differ in the scope of the "dense-set" or the construction of intersection graph. We introduce various kinds of quality measures for clustering in terms of local quality, global quality and validation of clusters. We generate simulated data of various hypergraphs assuming a set of underlying clustering structures. With these data, we compare the results and performances of our algorithms with the results of the other method, hMETIS [KK98a], using the quality

measures. We also apply our algorithms to the DIP (the Database of Interacting Proteins) dataset of PPI networks [XSD+02], and compare the results of our algorithm with the results of the existing methods, MCODE[BH03], CFinder[APF+06], PEWCC[EZB12], and NCmine[TK16]. Chapter 6 concludes the thesis by providing a summary of the work and the evaluation results. It also suggests prospects for future research.

Chapter 2

Preliminaries

2.1 Basic graph theory

Graph theory is the study of graphs, which are models for objects with pairwise relations. In this thesis, we develop a number of clustering algorithms based on the graph theory. We first introduce the basic notions and terminologies in graph theory used in this research.

Definition 1. (Graph) *A **graph** G is an ordered pair $(V(G), E(G))$ consisting of a set $V(G)$ of vertices and a set $E(G)$ of edges, where each edge of G is an unordered pair of vertices of G.*

Given a graph G, a function $w_G : V(G) \to \mathbb{R}$ can be defined on a vertex set $V(G)$. We call $w_G(v)$ the *weight* of $v \in V(G)$. Together with weights on its vertices, G is said to be a *vertex-weighted graph* and denoted by (G, w_G). Given a graph G, let $w'_G : E(G) \to \mathbb{R}$ be the function defined on an edge set $E(G)$. We call $w'_G(e)$ the *weight* of $e \in E(G)$. Together with weights on its edges, G is said to be an *edge-weighted graph* and denoted by $(G, {w'}_G)$. A subgraph $G' = (V(G'), E(G'))$ of G is a graph with a vertex set $V(G') \subset V(G)$ and an edge set $E(G') \subset E(G)$. A subgraph G' of G is said to be induced by a vertex set $S \subset V(G)$ if $S = V(G')$ and $E(G') = \{e | e \in V(G')$ and $e \in E(G)\}$. We denote by $G[S]$ the subgraph of a graph G if it is induced by $S \subset V(G)$. The number of edges incident to a vertex of a graph G is called the *degree* of v and denoted by $d_G(v)$. A set of vertices adjacent to a vertex of a graph G is called the *neighborhood* of v and

denoted by $N_G(v)$. The union of $N_G(v)$ and $\{v\}$ is denoted by $N_G[v]$. We mean a partition of $V(G)$ by a partition of G.

Definition 2. (Interval graph) *Let $\mathcal{I}$=$\{I_1,\ I_2,\ \ldots,\ I_n\}$ be a set of closed intervals of real lines. The **interval graph** G of $\mathcal{I}$, denoted by $G_{\mathcal{I}}$, is defined as a graph $(V_{\mathcal{I}},\ E_{\mathcal{I}})$ where $V_{\mathcal{I}} = \{v_1,\ \ldots,\ v_n\}$ and $\{v_i,\ v_j\} \in E_{\mathcal{I}}$ if and only if $I_i \cap I_j \neq \emptyset$.*

Definition 3. (Line graph) *For a given graph G, its **line graph**, denoted by $L(G)$, is a graph such that the set of vertices is $E(G)$ and two vertices $e_i, e_j \in E(G)$ are joined by an edge if and only if $e_i \cap e_j \neq \emptyset$.*

Definition 4. (Clique) *A **clique** of a graph G is a set of vertices $C \subset V(G)$ any two of which are adjacent. A **maximal clique** in G is a clique which is not a subset of a larger clique. A **largest clique** in G is a clique which has the maximum size among all the cliques in the graph.*

Definition 5. (Independent set) *A set of vertices is said to be an **independent set (IS)** if no two vertices in the set are adjacent. In a vertex-weighted graph (G, w_G), **maximum weight independent set (MWIS)** is an independent set of which the sum of weights on vertices is greatest among the sums of weight of all the independent set*

Definition 6. (Intersection graph) *An **intersection graph** $I_{\mathcal{F}}$ of a family of the sets $\mathcal{F} = \{F_1, F_2, ..., F_k\}$ is a graph in which a set of vertices is the family $\mathcal{F}$ and if two vertices $F_i, F_j \in \mathcal{F}$ are joined by an edge if and only if $F_i \cap F_j \neq \emptyset$.*

Definition 7. (Hypergraph) *A **hypergraph** $H = (V(H), E(H))$ is defined as a set of vertices $V(H)$ and a set of hyperedges $E(H)$ where each **hyperedge** is a subset of $V(H)$.*

Hypergraphs can be thought as generalizations of graphs. We may give a function $w'_H : E(H) \to \mathbb{R}$ from the hyperedge set of a hypergraph to the set of real numbers, similarly to the graph case. A *sub-hypergraph* $H' = (V(H'), E(H'))$ of H is a hypergraph with a vertex set $V(H') \subset V(H)$ and an edge set $E(H') \subset E(H)$. A sub-hypergraph H' of H is said to be *induced* by a vertex set $U \subset V(H)$ if $U = V(H')$ and $E(H') = \{e | e \in E(H)$ and $e \in V(H')\}$. A degree of a vertex v in H, denoted by $d_H(v)$, is defined as the number of hyperedges which contains the vertex v, i.e., $d_H(v) = |\{e \in E(H) | v \in e\}|$. We mean a partition of $V(H)$ by a partition of H.

Similarly for the line graph of a graph, the *line graph of a hypergraph* can be defined as follows: given a hypergraph H, its line graph, denoted by $L(H)$, is a graph such that the set of vertices is $E(H)$ and two vertices $e_i, e_j \in E(H)$ are joined by an edge if and only if $e_i \cap e_j \neq \emptyset$.

2.2 Terminologies in genetics

Genome-Wide Association Studies (GWAS)

GWAS is an attempt to identify the associations between genetic risk factors and diseases in the population by analyzing the DNA sequence variations across the human genome. The important goals of GWAS include making predictions about the risk of disease and identifying the biological basis of genetic effects for developing new treatment and prevention of disease [BM12].

Single Nucleotide Polymorphism (SNP)

An SNP is a variation at a single base position in a DNA sequence among individuals [BM12]. It has been predicted that there are more than 10 milion SNPs in the human genome for minor allele frequency (MAF; the frequency of the least

frequent allele) at 1% [KN$^+$01]. SNP data have been analyzed for gene-trait association studies as well as population genetic research.

Linkage Disequilibrium (LD)

LD is the non-random association between alleles at different loci (positions on a chromosome; the singular form is 'locus'). Suppose there are alleles A and a at an biallelic marker and B and b at another marker. Let the allele frequencies of A and B be p_A and p_B. Similarly let p_{AB} be the frequency with both A and B occuring together in the same gamete. The LD coefficient D is defined as

$$\mathrm{D} = p_{AB} - p_A p_B.$$

Since the range of the LD coefficients depends on the frequency of the alleles, Lewontin [Lew64] suggested normalizing D as follows:

$$\mathrm{D}' = \mathrm{D}/\mathrm{D}_{\min}$$

where

$$\mathrm{D}_{\min} = \begin{cases} \max\{-p_A p_B, -(1-p_A)(1-p_B)\} & \text{when } \mathrm{D} < 0 \\ \min\{p_A(1-p_B), (1-p_A)p_B\} & \text{when } \mathrm{D} > 0 \end{cases}$$

However, a high value of D$'$ cannot guarantee that one locus can predict the other with high accuracy. The correlation coefficient r or r^2 can be an alternative to D$'$, and r is defined as

$$r = \frac{\mathrm{D}}{\sqrt{p_A p_B (1-p_A)(1-p_B)}}$$

Recombination

Recombination is a process in which pieces of DNA are broken and recombined to

produce new combinations of alleles. This recombination process creates genetic diversity at the level of genes that reflects differences in the DNA sequences of different organisms. Recombination events preferentially occur in specific regions called *recombination hotspots*.

Positive Selection

Positive selection is the process in which beneficial genetic variants tend to become more frequent in populations. Understanding the traits of positive selection provide insight into how humans have evolved and the emergence of various diseases today [SSF+06].

Chapter 3

Clustering of genomic data

3.1 Background

Understanding the patterns of linkage disequilibrium (LD) across the human genome, especially its block-like structures, has interested many researchers in genetics because it provides useful insight for disease association mapping and population genetics [GBH+03, SVF+07, Sla08, DRS+01, RCB+01b, JKN01, BFMD05, TMP+03, WAZ+02]. These blocks are called LD blocks or haplotype blocks. It has been observed that LD block regions can vary in size, and strong LD extends over each block region until it breaks down abruptly possibly due to recombination hotspots or population genetic phenomenon [JKN01, SS05]. Recombination, mutation, selection, or other evolutionary history can affect LD block patterns in the aspects of location and length of blocks [MMH+04]. Identifying LD blocks, therefore, can provide evidential groundings for population genetic arguments and necessary information for the design and analysis of genetic association studies for complex diseases [REW+04, WAZ+02, WKE+10].

Researchers have proposed several methods to determine LD blocks from population data consisting of independent unrelated individuals [PRFP08, GSN+02, WAZ+02, BFMD05, TGP14]. Pattaro [PRFP08] developed an method, called MATILDE, to identify LD blocks based on LD measure either D' [Lew64] or r^2 using a MCMC algorithm and used the partitioning results in genetic association analysis. Gabriel et al [GSN+02] showed that the human genome can be partitioned

into LD blocks within which little evidence of recombination and less haplotype diversity is observed. They proposed a method, called the confidence interval (CI) method, to define haplotype blocks using a 95% confidence bound for an LD measure D' [Lew64]. Wang et al [WAZ+02] suggested a different approach to defining LD blocks using the four-gamete test (FGT) that identifies the occurrence of past recombination events. Both methods, CI and FGT, are implemented in the software package Haploview [BFMD05] which is a visualization and analysis tool for haplotype patterns. In Haploview, LD blocks can be recognized by an additional algorithm called "Solid Spine (SS)" with which an LD block consists of all the SNPs that are in strong LD with the first and last SNPs in the block (Haploview documentation). Recently, Taliun et al. [TGP14] developed an algorithm called MIG++, with increased computational efficiency incorporating an incremental computation strategy and estimation methods of approximated confidence interval for D' [Zap11] applying the LD block definition of Gabriel et al. [GSN+02] to extensive high-throughput datasets such as whole-genome sequencing data; a version of MIG++ is now implemented in PLINK [CCT+15]. Another computationally improved LD block construction algorithm named S-MIG++, also using the definition of Gabriel et al. [GSN+02], adopts a two-step approach: the estimation step finding the upper limits of the haplotype blocks by sampling a fraction of SNP pairs and the refinement step determining exact haplotype boundaries [TGLP16].

Nowadays, high-throughput genotype data such as genome-wide SNP array data or whole genome/exome sequencing data are collected and analyzed for gene-trait association studies [Man10, WLZ15]. An efficient approach for the analysis of high-throughput genomic data and a good alternative to single-SNP methods is to combine the effects of multiple SNPs on disease phenotype, [NS04, Pan09, WKE+10, NRV+11]. However, to apply multi-SNP analysis methods, it is ne-

cessary to specify a set of SNPs to be combined prior to performing multi-SNP tests. To specify these SNP sets, LD block information can be utilized in combination with gene or pathway information [WKE+10, ZGS+11]. For some multi-SNP tests, higher LD among combined SNPs can improve the power of the test, but having fewer SNP-set analysis units and independent global null hypotheses that correspond to each of the units is preferred as a simple way to reduce type I error inflation [YKB15, DLS14]. Yoo et al. [YKB15] proposed and evaluated a gene-based multi-SNP combination method that combines the effect of SNPs in "LD bins" using CLQ, a *clique*-based SNP clustering algorithm.

In this thesis, we develop a new algorithm called Big-LD that adopts a wider sense LD block definition - a block region based on the LD bin clusters. To do this we utilize LD bins constructed using CLQ-D, a modified version of the previous LD bin construction algorithm CLQ [YKB15, YSP+17], and consider the intervals defined by the physical positions of the starting and ending SNPs of each LD bin. Each interval corresponding to an LD bin represents the physical range of LD associated with SNPs in that bin. In the Big-LD algorithm, we model the relationships between intervals by a graph with vertices that represent the LD bin intervals and edges that represent the overlap between LD bin intervals. By applying an algorithm that partitions the vertex set of the interval graph into mutually exclusive sets by selecting an independent set with the maximum weight, Big-LD automatically determines the number of partitions to be formed into LD blocks. Using data from the 1000 Genomes Project and the HapMap Phase III of East Asian populations, we show that the LD blocks found by Big-LD are larger than the blocks produced by the existing methods MATILDE, Haploview, MIG++ or S-MIG++. Furthermore, using data of class II regions of MHC with true recombination hotspot information experimentally determined in a prior study [JRN00, JKN01], we show that, compared to

the existing methods, the LD block boundaries obtained by Big-LD coincide more closely with recombination hotspot positions.

3.2 Algorithms

3.2.1 CLQ algorithm

Suppose that the SNP sequence data consist of genotypes of a total of n SNPs in m individuals. Let S=$\{s_1, s_2, \ldots, s_n\}$ be the set of SNPs and $y(s_i)$ be the physical position (in base pair; bp) of s_i in the chromosome. We may assume that $s_1, s_2, \ldots, s_n$ are arranged so that $y(s_1) < \cdots < y(s_n)$. We also assume an additive genetic model such that the genotype of each individual for each SNP is the count of the risk allele (usually the minor allele), i.e. either of 0, 1 or 2. Each genotype vector of SNP s_i of m individuals is denoted as X_i. Then the pairwise LD measure r_{ij}^2 between two SNPs s_i and s_j can be obtained by squaring the Pearson correlation coefficient r_{ij} between additive genotypes X_i and X_j. We can construct an edge-weighted complete graph G with the given SNP set S as the vertex set and the edge set E=$\{\{s_i, s_j\} \mid s_i,\ s_j \in V,\ \ i \neq j\}$ with $w_E(\{s_i, s_j\}) = |r_{ij}|$ for $\{s_i, s_j\} \in E$. Then the SNPs and their LD structures are represented by the graph $G = (S, E, w_E)$.

Overall, LD between two SNPs tends to decrease as the physical distance between them increases [Tak82, CLM99, Ott00]. However, this relationship does not occur monotonically, and a group of strongly correlated SNPs can be formed by non-consecutive SNPs. We call a group of strongly correlated SNPs an *LD bin* within which every SNP is highly correlated with the other SNPs.

We developed an algorithm called CLQ that partition SNPs into LD bins using a greedy clique partitioning method [YKB15]. The CLQ algorithm starts by mod-

eling SNP set data by a graph in which vertices are SNPs and edges between two SNPs s_i and s_j exist whenever $|r|$ between them exceeds a given threshold. Then, by the Bron-Kerbosch algorithm [BK73, ELS10], all maximal cliques in the graph are identified and among them the largest clique is taken as the first LD bin. We then remove SNPs already taken as the LD bin, and again choose the next largest clique as the next LD bin among remaining maximal cliques. Repeating this procedure, the CLQ algorithm partitions the SNPs into a set of LD bins that are actually cliques in the graph.

The CLQ algorithm is as follows:

Step 1. To find high-correlated groups of a given graph $G = (S, E, w_E)$, delete edges with the relatively small weights. To be more specific, given a threshold θ, an edge $\{s_i,\ s_j\}$ satisfying $w_E(\{s_i, s_j\}) < \theta$ is to be deleted from G. Let $G_1 = G$ with $S_1 = S$ and $E_1 = E$ be the graph obtained by deleting those edges.

Starting with B_1, iterate the selection of the k^{th} cluster B_k found from the graph G_k in steps 2 to 4.

Step 2. For each vertex in G_k, find the maximal cliques that contain the vertex using the Bron-Kerbosch algorithm implemented in igraph package [BK73, ELS10] and select the largest clique of maximal cliques found for all vertices. Proceed to **Step 5** if there is no maximum clique with at least two vertices. Otherwise, proceed to the next step.

Step 3. Apply the Wang and Elston SNP recoding algorithm [WE07] (Appendix A) to the maximum cliques chosen in **Step 2**. If all pairwise correlations between SNPs in the clique can be recoded to be positive, then take the SNPs corresponding the chosen clique as the cluster B_k. If negatively correlated SNPs still exist after the coding correction algorithm has been applied to this clique, discard the chosen clique and select the next largest. If there are multiple cliques in G_k with the largest

size and all SNPs can be recoded to be positively correlated, chose the one with the largest sum of absolute correlation. Repeat application of the recoding algorithm until B_k is determined. If there is no clique with at least two vertices that can be recoded to have all positive correlations, proceed to Step 5.

Step 4. Remove SNPs in B_k from S_k and denote it as $S_{k+1} = S_k \backslash B_k$. Let G_{k+1} be a subgraph of G_k induced by S_{k+1}. Iterate Steps 2$\sim$4 unless the condition to proceed to Step 5 is met or all SNPs are assigned into a cluster.

Step 5. If there is no maximum clique with at least two vertices in G_k, the SNPs in V_k will be partitioned into singleton clusters. In this way all the SNPs are assigned into clusters $B_1, \cdots, B_l$ for some $l \leq n$. Then $V_1 = \bigcup_{k=1}^{l} B_k$ where $B_j \bigcap B_k = \emptyset$ for $j \neq k$ and $B_k \neq \emptyset$ for $k = 1, \cdots, l$.

3.2.2 CLQ-D: construction of LD bins

For the new LD block partition algorithm, we modify the original CLQ LD bins selection to give priority to bins where SNPs are relatively close in physical proximity. To do this, we develop CLQ-D as a modified version of CLQ incorporating a marker-density function of cliques defined using the size k of the clique and the range of physical positions of the SNPs in the clique. The marker-density function D of a clique Q with size k is defined as $D(Q) = \frac{k}{b-a}$ if $k > 1$, where a and b are the minimum and the maximum, respectively, of the bp positions of the SNPs in Q, and $D(Q) = 0$ if $k = 1$. When choosing an LD bin among the set of maximal cliques in each iteration of the greedy algorithm, we give the priority to the clique with the maximum marker-density D. The details of the newly developed CLQ-D algorithm are as follows:

Step 1. To find high-correlated groups of a given graph $G = (S, E, w_E)$, delete edges with the relatively small weights. To be more specific, given a threshold θ, an

edge $\{s_i,\ s_j\}$ satisfying $w_E(\{s_i, s_j\}) < \theta$ is to be deleted from G. Let G' be the graph obtained by deleting those edges.

Step 2. Find the family of all the maximal cliques $Q = \{Q_1, Q_2, \ldots, Q_q\}$ of G' using the Bron-Kerbosch algorithm [BK73, ELS10] implemented in the R *igraph* package [CN06].

Step 3. If a clique in Q has two consecutive SNPs with distance between them is too far, say above a threshold ρ, divide the clique into two cliques so that the distance between two consecutive SNPs is within ρ. Update Q with these cliques in which the distance between two consecutive SNPs is at most ρ.

Step 4. Calculate the density of each clique in Q. Choose $i \in \{1, 2, \cdots, m\}$ such that the density $D(Q_i)$ is the maximum over Q and take Q_i as a first LD bin B_1 and let $Q^{(1)} = \{Q_1 \backslash B_1,\ \cdots,\ Q_m \backslash B_1\}$ where $Q_j \backslash B_1$ $(j = 1, 2, \cdots, q)$ is the set of elements in Q_j and not in B_1. Since a subset of a clique is also a clique, $Q^{(1)}$ is still a family of cliques.

Step 5. Apply the same procedure as Step 4 to $Q^{(1)}$ to obtain $Q^{(2)}$ and LD bin B_2. Repeat this procedure until all the cliques in $Q^{(t)}$ for some t are singletons. In this way we obtain $B_1, \ldots B_t$.

Step 6. Take the cliques in $Q^{(t)}$ that are singleton LD bins and add them to the set of the previously chosen LD bins $B_1, \ldots, B_t$. In this way all the SNPs are assigned into mutually exclusive LD bins $B_1, B_2, \ldots, B_l$ so that $S = \bigcup_{i=1}^{l} B_i$ and $B_i \cap B_j = \emptyset$ for $i \neq j$.

A heuristic procedure to reduce the computational time to find maximal cliques

For a graph *G* with *n* vertices, the algorithm finds all maximal cliques in time $O(dn3^{d/3})$ where the value *d*, the *degeneracy* of a graph *G,* is the smallest value

such that every nonempty subgraph of G contains a vertex of degree at most d [ELS10]. Since the runtime grows exponentially as the degeneracy increases, running CLQ-D for a non-sparse graph with many vertices may require massive computation time. The SNP sequence data sometimes have such regions that correspond to graphs with large edge-density (defined as the ratio of the number of edges in a graph over the number of all possible edges in a graph) when the LD extends over a large region. Therefore, we incorporate a heuristic procedure in CLQ-D to reduce computational time when the graph has many vertices and is not sparse.

To measure sparsity of a graph, we use the concept of a k-core of a graph and the coreness of a vertex. A *k-core* of a graph G is a maximal subgraph in which every vertex has at least degree k, and the *coreness* of a vertex is k if it belongs to a k-core of G, but not to any $(k+1)$-core. If there exists a k-core of G, the degeneracy of G is at least k. For CLQ-D, we decide that the graph is not sparse if the number of vertices is more than 500, and either the median and the maximum of the corenesses of the vertices in the largest component (a *component* is a subgraph in which any two vertices are connected by a sequence of edges) of the graph is greater than 80 and 100, respectively, or the 75^{th} percentile and the maximum of the corenesses are both greater than 100. If the graph is not sparse, we select the one having the highest edge-density among the subgraphs each of which is induced by a vertex with at least top 30% degree and its neighbors, and delete it from the graph. We repeat this procedure in a greedy manner until the graph induced by the remaining vertices is not sparse. If the remaining graph reaches the sparsity criteria, then we go to Step 2.

3.2.3 Big-LD: construction of LD blocks

Let $H = \{B_1, B_2, \ldots, B_l\}$ be the set of LD bins obtained by the CLQ-D algorithm (Figure 3.1B). Interval representation of H can be formulated from the physical positions (bp) of SNPs. For each LD bin $B_i = \{s_{i_1}, s_{i_2}, \ldots, s_{i_{k_i}}\}$ where $y(s_{i_1}) < y(s_{i_2}) < \cdots < y(s_{i_{k_i}})$, we can define an interval corresponding to B_i as $I_i = [y(s_{i_1}), y(s_{i_{k_i}})]$. We then obtain a set of intervals $\mathcal{I} = \{I_1, I_2, \ldots, I_l\}$ such that I_i corresponds to B_i for each $i = 1, \ldots, l$. (Figure 3.1C).

Each interval corresponding to an LD bin represents the range of the LD associated with SNPs in that bin. When some LD bin intervals overlap one another, we may infer that the LD extends over the combined region of these overlapping intervals. In this sense, we assume that a set of LD bin intervals that overlap each other are more likely to be included in the same LD block. Based on this assumption, we construct an interval graph which reflects the intersection patterns of the range of LD bins and cluster them into an optimal partition where the LD within the same block remains strong and the LD between the SNPs in different blocks is relatively low. To do this, first define an interval graph $G = (V, E)$ of $\mathcal{I}$ with the vertex set $V = \{v_1, v_2, \ldots, v_l\}$ and the edge set $E = \{\{v_i, v_j\} | I_i \cap I_j \neq \emptyset\}$ (Figure 3.1D).

A set of mutually overlapping intervals corresponds to a clique in the interval graph G, and the union of the intervals corresponding to the clique forms an interval that could be considered as an LD block since strong LD extends over these overlapping intervals. By an appropriate clique partitioning of the interval graph G, we can find an optimized LD block partitioning of the genomic data. We perform this clique partitioning using a greedy algorithm which finds a maximum weight independent set of a graph which is updated at each iteration by taking the cliques in the current interval graph as the vertices and joining two vertices by an edge whenever

the intervals corresponding to the two vertices overlap in the current interval graph. For the vertex weights of this graph, we assign the number of SNPs in the clique corresponding to the vertex (Figure 3.1E). In this weighted graph constructed using all possible cliques of the interval graph, we choose an independent set of vertices such that the sum of weights assigned to its elements is the maximum. Each interval region corresponding to a vertex in this set is recruited as an LD block, and LD blocks selected in this way would not overlap each other since the vertices in an independent set do not have edges between them (Figure 3.1F). Once we recruit LD blocks from one iteration of the greedy algorithm, some SNPs could remain that are not covered by the intervals corresponding to the independent sets we have chosen so far. We apply the same process to the graph of updated intervals after deleting the chosen SNPs from the previously selected intervals. The detailed steps of Big-LD algorithm are as follows:

Step 1. Find all the cliques, say $C_1, C_2, \ldots, C_M$, in an interval graph $G = (V, E)$ of $\mathcal{I}$ by using the algorithm of Tsukiyama et al. [TIAS77] implemented in the R igraph package [CN06]. (If maximum coreness of G is greater than 10, we find all maximal cliques instead cliques by using the Bron-kerbosh algorithm [BK73, ELS10] to avoid exponential increase of the computational time and memory usage.)

Step 2. For each $i \in \{1, ..., M\}$, since the intervals corresponding to the vertices in C_i are mutually overlapping, the union of those intervals forms an interval which we denote by J_i.

Step 3. Take an interval graph of $\mathcal{J}^{(1)} = \{J_1^{(1)}, ..., J_M^{(1)}\}$ with the vertex set $V^{(1)} = \{v_1^{(1)}, ..., v_M^{(1)}\}$. We assign a weight to vertex $v_i^{(1)}$ as the number of all SNPs which belong to the LD bins corresponding to clique C_i for each $i = 1, ..., M$ to obtain a weighted interval graph.

Step 4. By using an algorithm to find a maximum weighted independent set of a

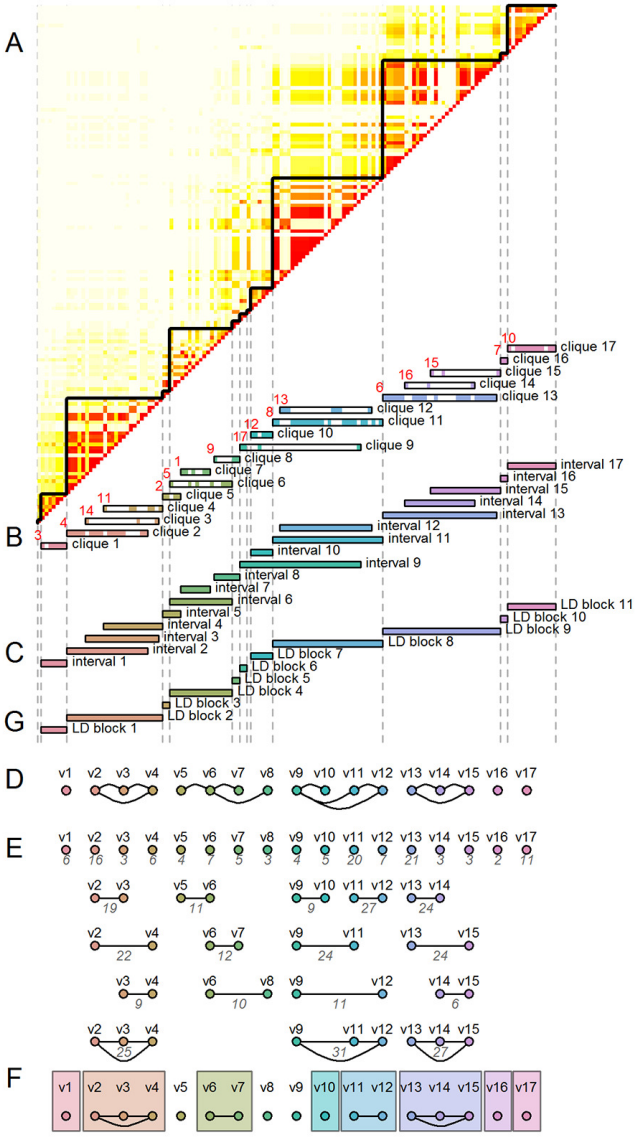


Figure 3.1: An illustration of CLQ-D and BIG-LD algorithm applied to a region in chromosome 22 (chr 22: 20,197,335∼20,277,113) using 1000G dataset. (A) An LD heatmap of 141 SNPs in this region. (B) LD bins constructed using CLQ-D algorithm in this region. The positions of SNPs in each LD bin region are shown in color in each strip. The numbers in red color show the order of LD bins selected by the greedy process of CLQ-D. (C) Interval representation of LD bins obtained in B. (D) The interval graph G of the set of intervals given in C. Each vertex corresponds to LD bin in B and each edge corresponds to the overlapping relationship between two intervals. (E) All possible cliques in G. The weight on each interval clique (grey colored number) is obtained as the total number of SNPs in the union of LD bins that correspond to the vertices of each clique in G. (F) The result of the first iteration of BIG-LD algorithm. For each chosen clique in G, union of intervals which correspond to vertices in the clique is taken as an LD block. (G) The final LD blocks obtained by BIG-LD algorithm.

vertex weighted interval graph (the details of the algorithm are stated in below, and its pseudo code can be found in Algorithm 3.1), find a maximum weight independent set $K^{(1)} = \{v_{i_1}^{(1)}, ..., v_{i_{k_1}}^{(1)}\}$ of the weighted interval graph obtained in Step 3. Note that the intervals $J_{i_1}^{(1)}, ..., J_{i_{k_1}}^{(1)}$ corresponding to the vertices in $K^{(1)}$ are mutually disjoint. We take $J_{i_1}^{(1)}, ..., J_{i_{k_1}}^{(1)}$as the first set of LD blocks obtained from the first iteration.

Step 5. Update each $J_i^{(1)}$, $i = 1, ..., M$ by removing the region taken as the LD block at the previous step, i.e. the difference set after removing the union of $J_{i_1}^{(1)}, ..., J_{i_{k_1}}^{(1)}$. It could become an empty set or a non-empty subinterval of $J_i^{(1)}$. Update $\mathcal{J}^{(1)}$ into $\mathcal{J}^{(2)}$ as the set of the resulting subintervals.

Step 6. If $\mathcal{J}^{(2)}$ is non-empty, then repeat the same procedure from Step 3 to Step 6 for $\mathcal{J}^{(2)}$ to obtain a set of LD blocks which are to be added to the previous set of LD blocks to form the current set of LD blocks. Repeat the iteration until the updated set of interval is empty. If it is empty set, stop and obtain the final set of LD blocks (Figure 3.1G).

An algorithm to find maximum weighted independent set (MWIS) of interval graph

Let $G = (V,\ E,\ w)$ be a vertex weighted interval graph and T be an interval representation of G. Also, we label V by the order of the left endpoints of the intervals such as $V = \{1, 2, \ \ldots, \ n\}$.

The construction of the maximum weighted independent set (MWIS) starts from the low order vertices to the high order vertices. First, we obtain P_i=$\{u \in V | u < i,\ u \notin Adj\ (i)\}$ for each vertex $i \in V$ where $Adj\ (i)$ is the set of all adjacent vertices to i. Note that each vertex $u \in P_i$ is independent of vertex i, and $k \in P_i$ and $j \in P_k$ implies $j \in P_i$, i.e., j, k and i are all independent. We construct the set

Q_i=$P_i - \bigcup_{k \in P_i} P_k$ for each vertex $i \in V$ so that if $k \in P_i$ and $j \in P_k$, then $j \notin Q_i$. To find MWIS efficiently, we obtain the maximum cumulated weight sum $W(i)$ over the cumulated weight sums of the independent sets including i and independent vertices preceding i, and the pointer $H(i)$ indicating the independent and immediately preceding interval of interval i present in the independent set having $W(i)$ as the cumulated weight sum. The detailed steps are given as follows. Consider all independent sets L that contain vertex i and vertices that are independent of i and also of each other and their labels are less than i. Let K_i be the set satisfying that the sum of vertex weights is the maximum among L. Let $W\ (i)$ be the sum of vertex weights in K_i and $H(i)$ be the immediate predecessor of i in K_i considering the labels of the order of intervals. If there is no immediate predecessor of i in K_i, $H(i)$ does not exist. From 1 to n in V, we obtain $W(i)$ and $H(i)$ for each $i \in V$. Finally, if we take vertex x whose cumulated weight $W(x)$ is greatest of $W(i)$ for i= 1,. . .,n then we can construct MWIS K by recruiting the pointer $H(x)$, an immediate independent predecessor of i, recursively until we reach an interval with no independent predecessor.

The rules to divide the SNPs into sub-tasks

To apply the CLQ-D and the Big-LD algorithms to big data with many SNPs, we can partition the entire region of the data into sub-regions of consecutive SNPs and apply the CLQ-D and Big-LD separately to reduce the computing time and memory. To do that, we need some strategy to efficiently combine the results of these sub-tasks. In the implementation of the Big-LD algorithm to deal with data with many SNPs, we predefine an upper limit for sub-task size (λ), the number of SNPs in a sub-region, and divide the SNP sequence into sub-tasks for a given task size limit λ. The break points for sub-tasks are selected by examining each position between two

Algorithm 3.1: Algorithm to find maximum weighted independent set(MWIS) of interval graph

input : vertex set V of weighted interval graph and weight $w(v)$ of each vertex v
output: maximum weight independent set K

/* Procedure get P_i: */
for $i \in V$ **do**
  $P_i \leftarrow \emptyset$ **for** $u \notin Adj(i)$ **do**
    if $u < i$ **then**
      $P_i \leftarrow P_i \cup \{u\}$;

/* Procedure get Q_i: */
for $i \in V$ **do**
  for $j \in P_i$ **do**
    $Q_i \leftarrow P_i - P_j$;

/* Find maximum weight independent set K: */
$T \leftarrow V - \bigcup Q_i$ **for** $i \in V$ **do**
  if $Q_i = \emptyset$ **then**
    $W(i) \leftarrow w(i)$;
    $H(i) \leftarrow \emptyset$;
  else
    $W(i) \leftarrow \max_{j \in P_i} W(j) + w(i)$;
    $H(i) \leftarrow j$;
select $y \in V$ such that $W(y) = \max_{v \in V} W(v)$;
$K \leftarrow \emptyset$;
while $y \neq null$ **do**
  $K \leftarrow K \cup \{y\}$;
  $y \leftarrow H(y)$;
return K

consecutive SNPs on each chromosome in the sequence order. The first choices of break points are weak LD points where the r^2 value between an SNP chosen from the rightmost z_1 number of SNPs in the preceding sub-region and an SNP chosen from the leftmost z_1 number of SNPs in the following sub-region is always less than 0.5 for a given z_1 (in this study, always set to 200). In the searching process to find weak LD points between two consecutive SNPs in the sequence order, we skip to the next position if we find any pair of SNPs with r^2 greater than or equal to 0.5 while examining the SNP pairs closer to the position of interest with priority. If a sub-region partitioned in this way exceeds the size limit, we split it at some relaxed weak LD point closest to the center to minimize the number of SNP pairs which satisfy r^2 less than 0.5 between an SNP chosen from the rightmost $z_1/5$ number of SNPs in the preceding sub-region and an SNP chosen from the leftmost $z_1/5$ number of SNPs in the following sub-region. This procedure is repeated until each of the sub-regions has a size within the limit λ. If the algorithm fails finding the break points after scanning $5 \times \lambda$, then we divide the whole sequence into the sub-regions of size of $\lambda/2$.

Once sub-regions are decided in this way, the Big-LD algorithm is applied to each sub-region to obtain a tentative set of LD blocks. After we obtain the tentative set of LD blocks from all sub-tasks, we take all pairs of consecutive sub-regions and apply the Big-LD algorithm again for the SNPs included in the last LD block of the preceding sub-region and the first LD block of the following sub-region in order to determine final LD blocks at the boundary of sub-regions.

Big-LD execution for the data including rare/low frequency variants

Since the probability to detect LD is low when the minor allele frequency (MAF) of a SNP in consideration is low [Lew95, GHHW00], we basically developed the

Big-LD algorithm to construct LD blocks based on common SNPs. For the data including rare/low frequency SNPs, below a MAF threshold value (usually, 0.05 or 0.01), we devised a two-stage process in which an initial LD block construction is first performed using the SNPs of which MAF is at least a threshold value, followed by extra steps that assign rare/low frequency SNPs with the MAF below the threshold into initially constructed blocks or add new blocks. The detailed steps taken after initial LD block construction using common SNPs is done are as follows:

Step 1. Assign rare SNPs (MAF below the threshold) to an initially constructed LD block if the rare SNPs are located between the first SNP and the last SNP of the block.

Step 2. For each rare SNP located between two consecutive non-singleton LD blocks, say B_1 and B_2, obtain the proportion of the SNPs (common and rare) within each of B_1 region and B_2 region in strong LD with the rare SNP. Here, "strong LD" means $|r| > \theta$ (threshold for CLQ-D).

Step 3. If the proportions obtained in step 2 for all rare SNPs between B_1 and B_2 are all less than or equal to a threshold τ, execute Big-LD algorithm with all SNPs between B_1 and B_2 as inputs and add the new LD blocks found from it. The threshold τ is set to be 0.4 throughout this thesis, which has been chosen as the value close to the empirical mean of such values obtained from chromosome 22 common SNP data of the 1000G dataset.

Step 4. If there are rare SNPs located between blocks B_1 and B_2 with the proportions obtained in the step 2 exceeding the threshold τ, extend each of B_1 and B_2 to include that SNP, respectively. For the region remaining after extension of blocks, run Big-LD algorithm for the SNPs in that region as inputs and add the newly obtained LD blocks.

Step 5. If the extended regions of B_1 and B_2 in the step 4 overlap each other, run

Big-LD algorithm for the entire region of B_1, B_2, and the region between B_1 and B_2 as inputs and update LD blocks for the combined region.

3.3 Evaluation of Big-LD algorithm

3.3.1 Evaluation data

We conducted performance evaluations and comparisons with some existing methods using three datasets: 1000 Genomes Project phase 1 release 3 (1000G) with 286 individuals from JPT, CHB and CHS populations [C$^+$12]; HapMap phase III (HapMap) with 170 individuals from JPT and CHB populations [C$^+$10]; the class II region of the major histocompatibility complex (MHC) with 50 north-European British semen donors [JKN01].

To evaluate runtime, memory, and the effect of sub-region sizes for the implemented Big-LD algorithm, we used the 1000G dataset of chromosome 1 through 22 (13,288,240 non-monomorphic SNPs). To compare the Big-LD block partition results with the other methods and assess recombination hotspot estimation we used the 1000G phased genotype data of 75,582 SNPs in chromosome 22 (chr22: 16,050,612∼51,243,297) after trimming with a minor allele frequency threshold of 0.05 and excluding insertion/deletion(indel) polymorphisms. For the HapMap dataset, we used the phased genotype data of 13,994 SNPs in chromosome 22 (chr22: 16,180,203∼51,219,006) after applying the same pruning criteria as for 1000G.

The original MHC dataset contains unphased diploid genotypes for 264 SNPs and 22 indels in 84kb resequenced regions [JKN01]. In this study, we used a subset of 263 SNPs without missing data or any indels for comparison of the LD block partition methods and the recombination hotspot locations experimentally identified by sperm-typing analysis of the same individuals which have been reported in

[JKN01]. Since the original data are unphased diploid genotype data, the haplotypes of 263 SNPs were reconstructed by PHASE v2.1.1 program [SSD01, SS05] and these were used as the input data for various LD block partition methods and recombination hotspot estimation methods. The MHC dataset can be downloaded from `http://www.le.ac.uk/genetics/ajj/HLA/`.

3.3.2 Implementation and performance evaluation

We implemented the CLQ-D and Big-LD algorithms in R. For the modified Bron-Kerbosch algorithm [BK73, ELS10] that finds all maximal cliques in a graph, required in CLQ-D and Big-LD implementation, we used the R library *igraph* package [CN06] in which the modified Bron–Kerbosch algorithm is written in ANSI C. For the algorithm to find all cliques in the graph, we used the method of Tsukiyama et al. [TIAS77] implemented in R *igraph*. We apply CLQ-D algorithm adopting heuristic procedure for evaluation of the runtime and memory of the data sampled from the entire autosome data of 1000G. Unless noted otherwise, we run Big-LD without the heuristic procedure throughout the thesis. We set threshold values required to run CLQ-D as $\theta = 0.5$ and $\rho = 40$kb for entire evaluation. To decide the threshold θ, we evaluate the threshold θ for $|r|$ in terms of LD block construction results and choose a threshold value based on the results. We compared the mean r^2 values within constructed LD blocks and across consecutive LD blocks (Table 3.1). We found that the mean r^2 across consecutive blocks are almost same for the thresholds $\theta = 0.4$ and 0.5, excluding or including singletons, while the means increase rapidly when we go to the threshold $\theta = 0.6$ (from 0.155 to 0.165 and 0.139 to 0.149 respectively), and the mean r^2 across consecutive blocks excluding singletons is minimized for the threshold $\theta = 0.5$. Based on this result, we decided to recommend the threshold value $\theta = 0.5$ and obtain major results based

Table 3.1: Summary of the LD block partition result of chromosome 22 region (75,582 SNPs) of the 1000G dataset obtained by Big-LD using various threshold θ for $|r|$

θ	N. of blocks		Average r^2 within a block	Average r^2 across consecutive blocks		N. of SNPs Mean(SD)	Length of block(kb) Mean(SD)
	Excluding singletons	All	Excluding singletons Mean(SD)	Excluding singletons Mean(SD)	All Mean(SD)		
0.3	2200	3585	0.441 (0.264)	0.165 (0.155)	0.170 (0.195)	33.7 (68.2)	14043 (34184)
0.4	1935	3396	0.448 (0.243)	0.140 (0.129)	0.150 (0.163)	38.3 (71.2)	16080 (37333)
0.5	2000	3654	0.472 (0.243)	0.139 (0.128)	0.155 (0.156)	37.0 (69.2)	15427 (37001)
0.6	2050	4084	0.502 (0.246)	0.149 (0.133)	0.165 (0.16)	35.9 (65.9)	14897 (34612)
0.7	2175	4722	0.526 (0.246)	0.158 (0.146)	0.182 (0.167)	33.6 (61.7)	13835 (33114)
0.8	2479	6157	0.571 (0.255)	0.181 (0.16)	0.209 (0.182)	29.0 (53.7)	11772 (28198)
0.9	3044	8781	0.619 (0.271)	0.228 (0.189)	0.260 (0.226)	23.0 (43.9)	9087 (22223)

on $\theta = 0.5$. The implemented R functions are packaged into "BigLD" and can be downloaded from `http://github.com/sunnyeesl/BigLD`. All experiments and performance evaluations were performed using these R functions on a machine with Intel i7-6700 (3.4 GHz) CPU.

For genomic regions with many SNPs, Big-LD divides the task into several sub-tasks in which the number of SNPs is less than a predetermined limit (λ). We first assessed the effect of the number of SNPs in one sub-task on runtime and memory usage using various samples chosen according to MAF and inter-SNP distance (Figure 3.2). The run-time and memory required for each region without dividing them into sub-tasks grows exponentially with the number of SNPs (Figure 3.3 and Table 3.2). Next, we tested how an upper limit for sub-task partition affects the LD block partition results of Big-LD for sub-task sizes (λ) from 1000 to 3000 (increasing by 500) using the entire chromosome 22 data of 1000G dataset (Table 3.3 and 3.5). When the upper limit is smaller, the runtime and memory to partition the data is reduced with little difference in the block partition results. Since more than 99% of the LD block partition results using the upper limit of 1500 and 3000 for sub-task size are the same (Table 3.5) and the run-time and memory

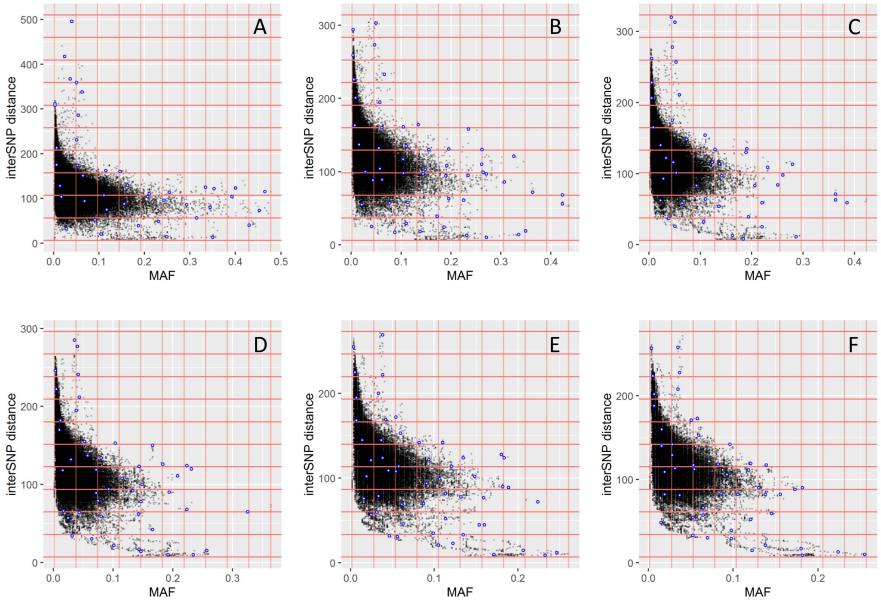


Figure 3.2: Two-dimensional distributions of the median MAF and the median inter-SNP distance of two consecutive non-monomorphic SNPs obtained for each window of fixed size shifting every 100 SNPs in chromosomes 1 through 22 of the 1000G dataset. Each plot from A to F corresponds to the fixed window size of 500, 1000, 1500, 2000, 2500, and 3000, respectively. We divide the domains of the two-dimensional distribution into 10×10 grid according to the ranges of the median MAF and the median inter-SNP distances, and then randomly choose a sample (window) per each non-empty grid. As a result, 43, 53, 45, 46, 54, and 49 samples for each window size 500, 1000, 1500, 2000, 2500, and 3000, respectively are chosen for evaluation of runtime and memory usage of Big-LD (blue dots)

Table 3.2: Runtime and memory usage of Big-LD algorithm for different number of SNPs (K=500, 1000, 1500, 2000, 2500, and 3000) without any sub-task partitioning evaluated over randomly selected sample regions from chromosome 1 through 22 of the 1000G dataset

	K	Number of samples	Min.	1st Qu.	Median	3rd Qu.	Max.	Mean (SD)
Memory (Mb)	500	43	46.7	99.9	1908.6	3668.0	10895.4	2205.8 (2332.5)
	1000	53	79.5	848.0	2351.2	4523.6	11072.2	2880.4 (2308.1)
	1500	45	258.6	1049.0	2417.2	4246.0	6412.9	2645.5 (1767.9)
	2000	46	767.3	2011.7	2750.0	3348.5	6341.1	2859.6 (1406.9)
	2500	54	814.1	2484.2	4174.5	5666.9	12370.8	4623.3 (2637.9)
	3000	49	832.7	2437.2	3490.1	4943.3	12192.5	3895.6 (2160.1)
Time (sec)	500	43	0.5	0.9	1.1	1.3	4.4	1.2 (0.6)
	1000	53	0.6	1.1	1.5	2.3	10.2	2 (1.6)
	1500	45	1.0	1.8	2.5	4.1	14.2	3.4 (2.6)
	2000	46	1.4	2.8	3.7	5.5	12.1	4.3 (2)
	2500	54	2.3	4.7	6.7	9.5	31.8	8.2 (5.9)
	3000	49	2.7	4.7	7.1	14.0	570.2	21.3 (80.4)

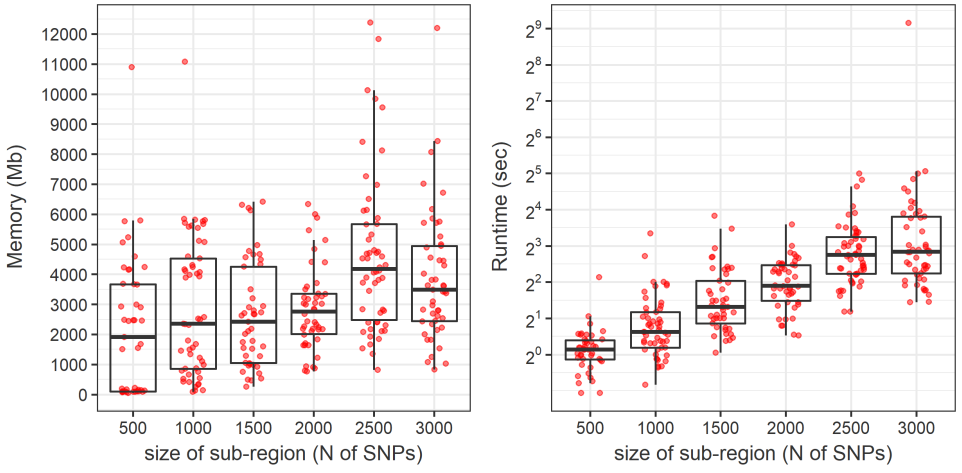


Figure 3.3: Boxplots of memory usage and runtime of Big-LD algorithm for samples of different number of SNPs (n =500, 1000, 1500, 2000, 2500, and 3000) without any sub-task partition. The boxplot shows the median, the first and third quartiles computed using Tukey's "hinges" and the end points of whiskers. The whiskers extend to the most extreme values no more than 1.5 times the interquartile range.

Table 3.3: Summary of the LD block partition result of chr 22 region (75,582 SNPs) obtained by Big-LD for different values of upper limit of sub-task size (λ)

Upper limit for sub-task size (λ)	1000G				HapMap			
	Runtime (sec)	Memory usage (MB)	N. of blocks		Runtime (sec)	Memory usage (MB)	N. of blocks	
			Excluding singletons	All			Excluding singletons	All
1000	273.18	2731.63	2002	3642	14.06	55.32	1254	2018
1500	291.92	3041.97	1999	3692	14.06	55.01	1254	2018
2000	290.10	3016.88	2000	3654	14.10	52.04	1254	2018
2500	346.84	3004.25	2015	3696	14.20	55.63	1254	2018
3000	358.12	4368.74	2016	3694	14.22	56.22	1254	2018

to run the same job is 81% and 70% respectively for the upper limit of 1500 instead of 3000, we decided to use 1500 as the upper limit for the number of SNPs in sub-tasks throughout the evaluation process. We also measured the runtime and memory usage of the sub-task partitioning stage for various sub-task size limit (λ) and for the data trimmed using two MAF threshold values of 0.05 and 0.01. The results show that mostly it takes more time to divide the sequence into sub-tasks with smaller λ values and to process the data including more low frequency variants (Figure 3.4 and Table 3.4). To assess the computational feasibility to apply the Big-LD for the entire genome sequence, we evaluated the runtime and the memory usage of the Big-LD algorithm applied to the entire 1000G dataset from chromosome 1 through 22 excluding only monomorphic SNPs. We averaged the runtime and the memory usage assessment over five repetitions of Big-LD executions for 1) Sub-task partitioning stage using SNPs with MAF $\geq$ 0.05, 2) Initial LD block partitioning stage using SNPs with MAF $\geq$ 0.05, 3) Entire LD block partitioning including the refinement steps for low frequency SNPs (Figure 3.5 and Table 3.6). The average runtime required to partition the entire dataset into sub-tasks of the limit size 1500 is 0.73 hours and the maximum memory usage is 3584Mb. The average

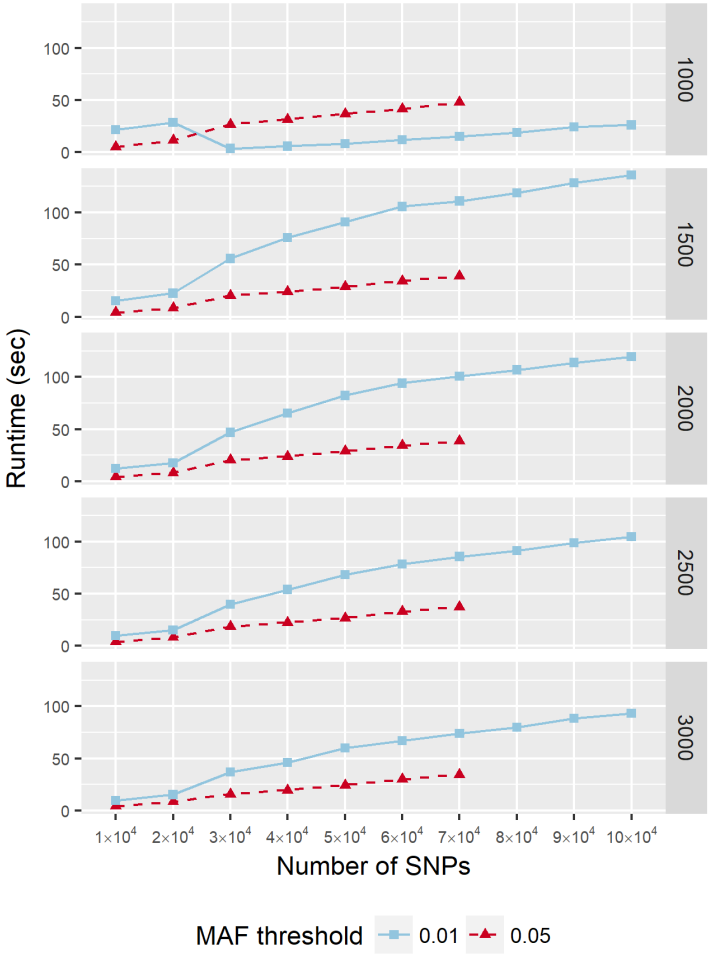


Figure 3.4: Average runtime of sub-task partitioning procedure of Big-LD using various values of the upper limit of sub-task size (λ =1000, 1500, 2000, 2500, and 3000) and two MAF threshold values (0.05 and 0.01) applied to sub-regions with various sizes of chromosome 22 of the 1000G dataset. We selected each sub-region in the center of the sequence for sizes from 10,000 to 100,000 SNPs (for MAF threshold 0.05, to 70,000 SNPs) by increasing 10,000 SNPs.

Table 3.4: Runtime and memory usage of sub-task partitioning procedure of Big-LD algorithm using various values of the upper limit of sub-task size (λ =1000, 1500, 2000, 2500, and 3000) and two MAF cut values (0.05 and 0.01) for chromosome 22 region of the 1000G dataset

MAF cut	N. of SNPs	λ	Runtime (sec) Mean (SD)	Memory (Mb) Mean (SD)
0.05	75,582	1000	51.59 (0.4)	1209.28 (0.08)
		1500	41.42 (0.23)	1208.68 (0.58)
		2000	41.36 (0.21)	1208.02 (1.23)
		2500	39.19 (0.28)	1208.65 (0.46)
		3000	36.78 (0.21)	1187.18 (0.76)
0.01	105,700	1000	28.48 (0.21)	1156.44 (0.17)
		1500	141.41 (2.22)	1161.14 (2.69)
		2000	125.04 (1.23)	1161.15 (2.77)
		2500	107.58 (0.87)	1159.96 (3.43)
		3000	96.33 (0.72)	1160.71 (2.57)

runtime and the maximum memory usage until initial LD block partitioning using only common SNPs (MAF $\geq$ 0.05; 5,429,840 SNPs) is 4.45 hours and 4144MB. It takes 5.83 hours to obtain LD blocks for the entire 1000G dataset of chromosome 1 through 22 excluding only monomorphic SNPs (13,288,240 SNPs) with maximum memory usage of 8828MB. We confirmed that runtime and memory usage increase linearly in most cases as the number of SNPs increases, which was expected, since Big-LD adopts a process of dividing the data into sub-tasks when massive SNP data are given (Figure 3.5).

We then compared the runtime and the memory usage of the Big-LD, S-MIG++, and MIG++ using the chromosome 22 data of 1000G dataset (Table 3.7). For the data including non-monomorphic SNPs, runtime of Big-LD is lowest (261.7sec), which is a significant improvement over the two existing methods (3148.5sec for MIG++ and 35678sec for S-MIG++). Meanwhile, for the SNPs with MAF$\geq$0.05, the runtime of MIG++ is lowest (114.6sec) followed by that of Big-LD (207 sec).

Table 3.5: Effect of the upper limit of sub-task size (λ) for Big-LD evaluated by overlap proportions of LD block partition results using two different size limits obtained for 1000G chromosome 22 data

Sub-task size (1)	Case	N. of blocks	Sub-task size (2)	Common block declaration criteria*						
				100%	99%	95%	90%	80%	70%	60%
1000	All	3642	1500	3558(0.98)	3558(0.98)	3562(0.98)	3567(0.98)	3570(0.98)	3572(0.98)	3575(0.98)
		3642	2000	3558(0.98)	3558(0.98)	3562(0.98)	3567(0.98)	3570(0.98)	3572(0.98)	3575(0.98)
		3642	2500	3565(0.98)	3565(0.98)	3568(0.98)	3574(0.98)	3577(0.98)	3580(0.98)	3583(0.98)
		3642	3000	3562(0.98)	3562(0.98)	3565(0.98)	3571(0.98)	3574(0.98)	3577(0.98)	3580(0.98)
	Excluding	2002	1500	1948(0.97)	1948(0.97)	1952(0.98)	1957(0.98)	1960(0.98)	1962(0.98)	1965(0.98)
	singletons	2002	2000	1948(0.97)	1948(0.97)	1952(0.98)	1957(0.98)	1960(0.98)	1962(0.98)	1965(0.98)
		2002	2500	1950(0.97)	1950(0.97)	1953(0.98)	1959(0.98)	1962(0.98)	1965(0.98)	1968(0.98)
		2002	3000	1950(0.97)	1950(0.97)	1953(0.98)	1959(0.98)	1962(0.98)	1965(0.98)	1968(0.98)
	Block width	906	1500	872(0.96)	872(0.96)	876(0.97)	881(0.97)	884(0.98)	885(0.98)	887(0.98)
	$\geq$5kb	906	2000	872(0.96)	872(0.96)	876(0.97)	881(0.97)	884(0.98)	885(0.98)	887(0.98)
		906	2500	874(0.96)	874(0.96)	877(0.97)	883(0.97)	886(0.98)	888(0.98)	890(0.98)
		906	3000	874(0.96)	874(0.96)	877(0.97)	883(0.97)	886(0.98)	888(0.98)	890(0.98)
	Block width	666	1500	640 (0.96)	640 (0.96)	643 (0.97)	647 (0.97)	650 (0.98)	651 (0.98)	652 (0.98)
	$\geq$10kb	666	2000	640 (0.96)	640 (0.96)	643 (0.97)	647 (0.97)	650 (0.98)	651 (0.98)	652 (0.98)
		666	2500	641 (0.96)	641 (0.96)	643 (0.97)	648 (0.97)	651 (0.98)	653 (0.98)	654 (0.98)
		666	3000	641 (0.96)	641 (0.96)	643 (0.97)	648 (0.97)	651 (0.98)	653 (0.98)	654 (0.98)
1500	All	3654	1000	3558(0.97)	3558(0.97)	3562(0.97)	3567(0.98)	3570(0.98)	3572(0.98)	3575(0.98)
		3654	2000	3654(1)	3654(1)	3654(1)	3654(1)	3654(1)	3654(1)	3654(1)
		3654	2500	3644(1)	3645(1)	3646(1)	3646(1)	3646(1)	3647(1)	3648(1)
		3654	3000	3641(1)	3642(1)	3643(1)	3643(1)	3643(1)	3644(1)	3645(1)
	Excluding	2000	1000	1948(0.97)	1948(0.97)	1952(0.98)	1957(0.98)	1960(0.98)	1962(0.98)	1965(0.98)
	singletons	2000	2000	2000(1)	2000(1)	2000(1)	2000(1)	2000(1)	2000(1)	2000(1)
		2000	2500	1992(1)	1993(1)	1994(1)	1994(1)	1994(1)	1995(1)	1996(1)
		2000	3000	1992(1)	1993(1)	1994(1)	1994(1)	1994(1)	1995(1)	1996(1)
	Block width	904	1000	872(0.96)	872(0.96)	876(0.97)	881(0.97)	884(0.98)	885(0.98)	887(0.98)
	$\geq$5kb	904	2000	904(1)	904(1)	904(1)	904(1)	904(1)	904(1)	904(1)
		904	2500	897(0.99)	898(0.99)	899(0.99)	899(0.99)	899(0.99)	900(1)	901(1)
		904	3000	897(0.99)	898(0.99)	899(0.99)	899(0.99)	899(0.99)	900(1)	901(1)
	Block with	662	1000	640 (0.97)	640 (0.97)	643 (0.97)	647 (0.98)	650 (0.98)	651 (0.98)	652 (0.98)
	$\geq$10kb	662	2000	662 (1)	662 (1)	662 (1)	662 (1)	662 (1)	662 (1)	662 (1)
		662	2500	657 (0.99)	658 (0.99)	659 (1)	659 (1)	659 (1)	660 (1)	660 (1)
		662	3000	657 (0.99)	658 (0.99)	659 (1)	659 (1)	659 (1)	660 (1)	660 (1)
2000	All	3654	1000	3558(0.97)	3558(0.97)	3562(0.97)	3567(0.98)	3570(0.98)	3572(0.98)	3575(0.98)
		3654	1500	3654(1)	3654(1)	3654(1)	3654(1)	3654(1)	3654(1)	3654(1)
		3654	2500	3644(1)	3645(1)	3646(1)	3646(1)	3646(1)	3647(1)	3648(1)
		3654	3000	3641(1)	3642(1)	3643(1)	3643(1)	3643(1)	3644(1)	3645(1)
	Excluding	2000	1000	1948(0.97)	1948(0.97)	1952(0.98)	1957(0.98)	1960(0.98)	1962(0.98)	1965(0.98)
	singletons	2000	1500	2000(1)	2000(1)	2000(1)	2000(1)	2000(1)	2000(1)	2000(1)
		2000	2500	1992(1)	1993(1)	1994(1)	1994(1)	1994(1)	1995(1)	1996(1)
		2000	3000	1992(1)	1993(1)	1994(1)	1994(1)	1994(1)	1995(1)	1996(1)
	Block width	904	1000	872(0.96)	872(0.96)	876(0.97)	881(0.97)	884(0.98)	885(0.98)	887(0.98)
	$\geq$5kb	904	1500	904(1)	904(1)	904(1)	904(1)	904(1)	904(1)	904(1)
		904	2500	897(0.99)	898(0.99)	899(0.99)	899(0.99)	899(0.99)	900(1)	901(1)

		904	3000	897(0.99)	898(0.99)	899(0.99)	899(0.99)	899(0.99)	900(1)	901(1)
	Block with	662	1000	640 (0.97)	640 (0.97)	643 (0.97)	647 (0.98)	650 (0.98)	651 (0.98)	652 (0.98)
	≥10kb	662	1500	662 (1)	662 (1)	662 (1)	662 (1)	662 (1)	662 (1)	662 (1)
		662	2500	657 (0.99)	658 (0.99)	659 (1)	659 (1)	659 (1)	660 (1)	660 (1)
		662	3000	657 (0.99)	658 (0.99)	659 (1)	659 (1)	659 (1)	660 (1)	660 (1)
2500	All	3696	1000	3565(0.96)	3565(0.96)	3568(0.97)	3574(0.97)	3577(0.97)	3580(0.97)	3583(0.97)
		3696	1500	3644(0.99)	3645(0.99)	3646(0.99)	3646(0.99)	3646(0.99)	3647(0.99)	3648(0.99)
		3696	2000	3644(0.99)	3645(0.99)	3646(0.99)	3646(0.99)	3646(0.99)	3647(0.99)	3648(0.99)
		3696	3000	3693(1)	3693(1)	3693(1)	3693(1)	3693(1)	3693(1)	3693(1)
	Excluding	2015	1000	1950(0.97)	1950(0.97)	1953(0.97)	1959(0.97)	1962(0.97)	1965(0.98)	1968(0.98)
	singletons	2015	1500	1992(0.99)	1993(0.99)	1994(0.99)	1994(0.99)	1994(0.99)	1995(0.99)	1996(0.99)
		2015	2000	1992(0.99)	1993(0.99)	1994(0.99)	1994(0.99)	1994(0.99)	1995(0.99)	1996(0.99)
		2015	3000	2015(1)	2015(1)	2015(1)	2015(1)	2015(1)	2015(1)	2015(1)
	Block width	913	1000	874(0.96)	874(0.96)	877(0.96)	883(0.97)	886(0.97)	888(0.97)	890(0.97)
	≥5kb	913	1500	897(0.98)	898(0.98)	899(0.98)	899(0.98)	899(0.98)	900(0.99)	901(0.99)
		913	2000	897(0.98)	898(0.98)	899(0.98)	899(0.98)	899(0.98)	900(0.99)	901(0.99)
		913	3000	913(1)	913(1)	913(1)	913(1)	913(1)	913(1)	913(1)
	Block width	670	1000	641 (0.96)	641 (0.96)	643 (0.96)	648 (0.97)	651 (0.97)	653 (0.97)	654 (0.98)
	≥10kb	670	1500	657 (0.98)	658 (0.98)	659 (0.98)	659 (0.98)	659 (0.98)	660 (0.99)	660 (0.99)
		670	2000	657 (0.98)	658 (0.98)	659 (0.98)	659 (0.98)	659 (0.98)	660 (0.99)	660 (0.99)
		670	3000	670 (1)	670 (1)	670 (1)	670 (1)	670 (1)	670 (1)	670 (1)
3000	All	3694	1000	3562(0.96)	3562(0.96)	3565(0.97)	3571(0.97)	3574(0.97)	3577(0.97)	3580(0.97)
		3694	1500	3641(0.99)	3642(0.99)	3643(0.99)	3643(0.99)	3643(0.99)	3644(0.99)	3645(0.99)
		3694	2000	3641(0.99)	3642(0.99)	3643(0.99)	3643(0.99)	3643(0.99)	3644(0.99)	3645(0.99)
		3694	2500	3693(1)	3693(1)	3693(1)	3693(1)	3693(1)	3693(1)	3693(1)
	Excluding	2016	1000	1950(0.97)	1950(0.97)	1953(0.97)	1959(0.97)	1962(0.97)	1965(0.97)	1968(0.98)
	singletons	2016	1500	1992(0.99)	1993(0.99)	1994(0.99)	1994(0.99)	1994(0.99)	1995(0.99)	1996(0.99)
		2016	2000	1992(0.99)	1993(0.99)	1994(0.99)	1994(0.99)	1994(0.99)	1995(0.99)	1996(0.99)
		2016	2500	2015(1)	2015(1)	2015(1)	2015(1)	2015(1)	2015(1)	2015(1)
	Block width	913	1000	874(0.96)	874(0.96)	877(0.96)	883(0.97)	886(0.97)	888(0.97)	890(0.97)
	≥5kb	913	1500	897(0.98)	898(0.98)	899(0.98)	899(0.98)	899(0.98)	900(0.99)	901(0.99)
		913	2000	897(0.98)	898(0.98)	899(0.98)	899(0.98)	899(0.98)	900(0.99)	901(0.99)
		913	2500	913(1)	913(1)	913(1)	913(1)	913(1)	913(1)	913(1)
	Block width	670	1000	641 (0.96)	641 (0.96)	643 (0.96)	648 (0.97)	651 (0.97)	653 (0.97)	654 (0.98)
	≥10kb	670	1500	657 (0.98)	658 (0.98)	659 (0.98)	659 (0.98)	659 (0.98)	660 (0.99)	660 (0.99)
		670	2000	657 (0.98)	658 (0.98)	659 (0.98)	659 (0.98)	659 (0.98)	660 (0.99)	660 (0.99)
		670	2500	670 (1)	670 (1)	670 (1)	670 (1)	670 (1)	670 (1)	670 (1)

* A(B): A is the number of common blocks between the blocks obtained by using the upper limit of sub-task size (1) and size (2). B is the relative frequency of A over the number of all blocks obtained by using the upper limit of sub-take size (1). An LD block is declared as a common block found using two sub-task size limit values if the relative percentage length of common area of two LD block results over the length of the entire LD block (in both results) is greater than the criterion.

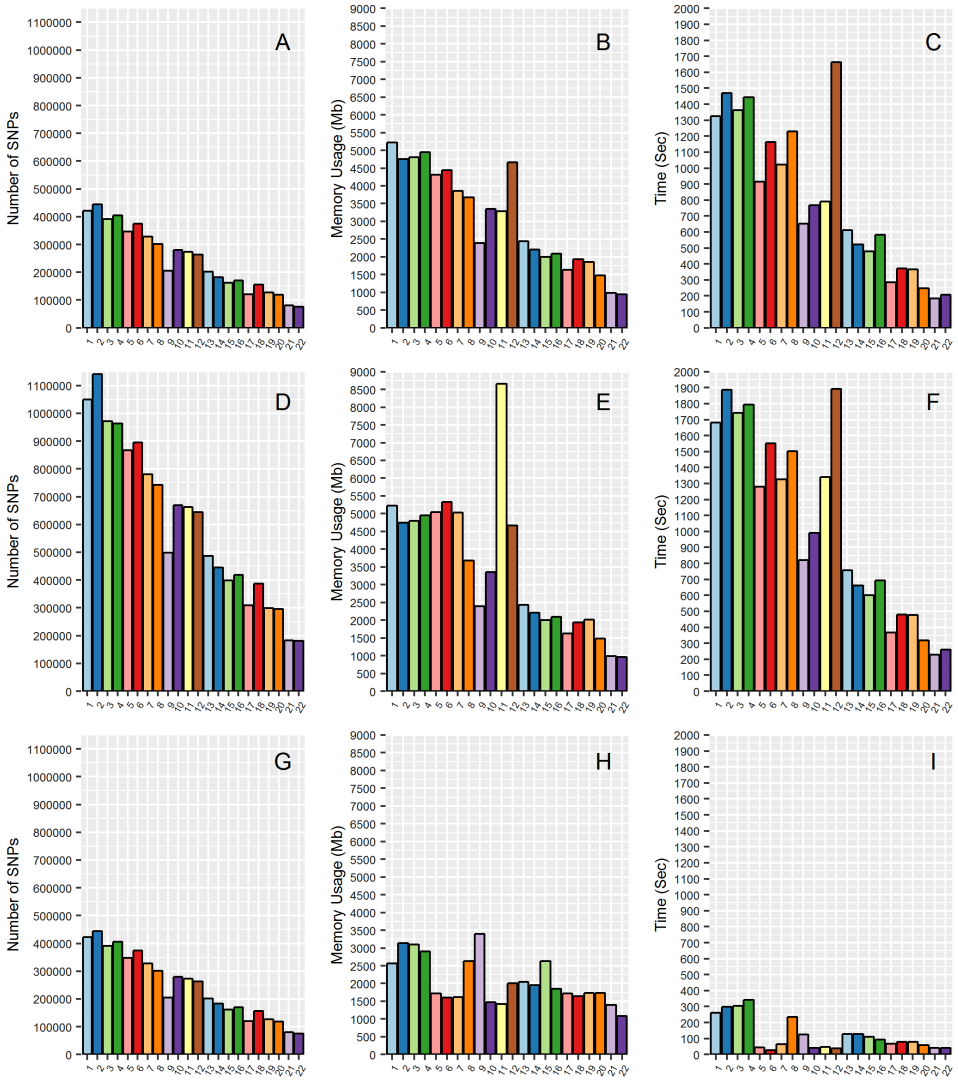


Figure 3.5: Runtime and memory usage of Big-LD for chromosome 1 through 22 of the 1000G dataset with the upper limit for sub-task size set as 1,500. (A) The number of SNPs after trimming with a MAF threshold of 0.05 for each chromosome, (B) The maximum memory usage measured for in A, (C) The runtime measured for data in A, (D) The number of SNPs after trimming only monomorphic SNPs for each chromosome, (E) The maximum memory usage measured for data in D, (F) The runtime measured for data in D, (G) The number of SNPs used in sub-task partitioning for each chromosome, (H) The memory usage of sub-task partitioning for data in G, (I) The runtime of sub-task partitioning for data in G.

Table 3.6: Runtime and memory usage of Big-LD algorithm for the data of non-monomorphic SNPs in chromosome 1 through 22 of the 1000G dataset with the upper limit of sub-task size (λ) of 1,500 and MAF threshold of 0.05

	Sub-task partitioning stage using SNPs with MAF≥0.05			Initial LD block partition stage using SNPs with MAF≥0.05 (Including Sub-task partitioning step)			Entire LD block partition for SNPs with MAF*		
Chr	N. of SNPs	Runtime (sec) Mean (SD)	Memory (Mb) Mean (SD)	N. of SNPs	Runtime (sec)** Mean (SD)	Memory (Mb) Mean (SD)	N. of SNPs	Runtime (sec)** Mean (SD)	Memory (Mb) Mean (SD)
1	421698	260.11 (3.77)	2567.99 (169.15)	421698	1126.12 (115.07)	4127.51 (14.74)	1050013	354.89 (10.79)	2418.78 (375.04)
2	444212	298.7 (0.59)	3133.62 (281.01)	444212	1271.62 (111.84)	3594.31 (10.35)	1141320	416.06 (28.26)	3869.7 (522.55)
3	391132	304.31 (1.36)	3095.55 (308.15)	391132	1199.91 (100.59)	3824.65 (16.58)	971719	378.4 (22.7)	4686.92 (390.06)
4	405345	342.3 (1.69)	2904.71 (328.01)	405345	1285.29 (102.11)	3845.28 (84.91)	963721	349.43 (20.54)	3604.27 (407.74)
5	347161	44.05 (0.13)	1714.58 (185.19)	347161	821.94 (80.06)	3403.56 (10.18)	866756	364.08 (17.87)	5043.32 (778.22)
6	374593	26.12 (0.23)	1601.44 (176.4)	374593	1071.7 (83.15)	3469.41 (8.91)	895195	385.85 (24.34)	5323.91 (462.87)
7	327658	64.45 (0.27)	1616.74 (132.05)	327658	926.11 (70.92)	3032.25 (18.66)	780368	304.82 (17.17)	5032 (920.53)
8	300957	235.78 (0.77)	2631.32 (183.03)	300957	1100.08 (82.65)	2889.49 (8.26)	743161	272.38 (17.91)	3021.03 (129.27)
9	204681	125.36 (1.42)	3401.54 (350.36)	204681	600.99 (32.61)	1917.02 (59.57)	498001	166.72 (10.15)	1852.17 (284.06)
10	280376	42.25 (1.55)	1478.02 (91.38)	280376	691.31 (51.37)	2694.04 (61.27)	669963	222.36 (13.06)	2197.11 (304.73)
11	273534	45.88 (0.95)	1419.76 (115.74)	273534	721.45 (47.55)	2562.23 (11.17)	662884	551.71 (17.44)	8657.66 (160.12)
12	263325	37.12 (0.76)	2003.45 (52.37)	263325	1695.40 (162.97)	3980.75 (28.07)	644113	230.15 (7.72)	3419.45 (111.11)
13	201555	127.96 (1.47)	2039.73 (220.55)	201555	549.90 (35.95)	1897.75 (6.54)	486334	146.58 (9.27)	1947.23 (275.44)
14	182599	126.77 (0.73)	1948.25 (210.12)	182599	481.66 (28.48)	1744.13 (11.26)	445332	140.19 (7.47)	1661.11 (560.84)
15	161910	110.81 (0.28)	2629.57 (85.28)	161910	443.15 (24.65)	1572.89 (8.03)	397938	121.85 (7.22)	1366.75 (349.85)
16	170315	94.82 (0.31)	1847.07 (198.32)	170315	534.98 (30.43)	1632.96 (8.12)	418929	109.9 (7.71)	1017.97 (313.41)
17	120946	68.83 (0.27)	1720.61 (177.38)	120946	249.8 (19.13)	1121.74 (97.08)	308521	81.85 (5.93)	730.1 (194.9)
18	156310	78.42 (0.39)	1646.62 (177.75)	156310	334.83 (20.68)	1509.35 (3.67)	386250	108.84 (6.38)	1166.86 (272.77)
19	127254	78.43 (0.49)	1731.75 (178.03)	127254	322.46 (24.16)	1464.63 (126.56)	299583	111.17 (3.37)	2016.13 (767.75)
20	118374	58.93 (1.35)	1738.84 (177.34)	118374	218.88 (15.82)	1166.84 (7.52)	294897	71.9 (2.76)	993.58 (290.27)
21	80323	42.06 (0.83)	1392.53 (82.35)	80323	169.85 (9.31)	821.54 (51.89)	182943	43.24 (0.87)	788.81 (235.89)
22	75582	41.42 (0.23)	1208.68 (0.58)	75582	187.64 (12.2)	1679.77 (842.25)	180299	51.74 (2.06)	966.7 (200.04)
Total†	5429840	2654.87 (12.51)	3583.81	5429840	16005.07 (1044.42)	4144.55	13288240	20989.18 (1275.83)	8828.06

* After initial LD block partition stage, SNPs with MAF ¡0.05 are assigned to existing blocks or newly formed blocks.

** Big-LD has been executed adopting CLQ-D implementing a heuristic procedure to reduce run-time.

† The average and standard deviation of the total runtime for chromosome 1 through 22, and the maximum memory usage of all executions.

Table 3.7: Runtime and memory usage of Big-LD, S-MIG++, and MIG++ for chromosome 22 region of the 1000G dataset

	1000G data (MAF $\geq$ 0.05)		1000G data (MAF $>$ 0)	
	Runtime (sec)	Memory (Mb)	Runtime (sec)	Memory (Mb)
Big-LD (1)	291.9	3042	322.3	620.1
Big-LD (2)*	207	940.4	261.7	1317.9
S-MIG++	6414.2	24	35378	25.4
MIG++	114.6	n/a	3148.5	n/a

* Results obtained by Big-LD adopting CLQ-D algorithm implementing a heuristic procedure to reduce runtime and memory usages.

We did not include Haploview and MATILDE in this comparison since the runtime for these programs is much longer than that of the other methods. For example, to run MATILDE for a set of only 1000 SNPs, it takes about 4.33 hours, and to run Haploview (CI, FGT, SS) for a set of 1500 SNPs, it takes about 0.17 hours.

3.3.3 Comparisons of Big-LD block partition results with pre-existing methods

Using chromosome 22 data of the 1000G dataset and HapMap dataset, we compared LD block partition results of Big-LD with those produced by MATILDE [PRFP08], Haploview (CI, FGT, SS) [BFMD05], MIG++ [TGP14] implemented in PLINK1.9 [CCT+15], and S-MIG++ [TGLP16]. Due to runtime and memory limitations of Haploview program, we applied the Haploview computation to data divided into windows of 1500 SNPs shifted every 1000 SNPs and combined the results with decision rules for the overlapped regions that give priority to the larger block among two results. For MATILDE, we divided the data into windows of 1000 SNPs and obtained results for each window. We also ran the program for 500 SNPs located around each of the break points between windows and then combined results for the 1000 and 500 SNP windows similarly to the rules to combine Haploview results.

In Table 3.8, we summarize several characteristics of the LD block partition results obtained by the Big-LD and other methods including the total number of blocks produced, the average length of blocks meaning the difference in the base pair (bp) position between the first and the last SNPs in the block, the average number of SNPs per block, the average r^2 over all the pairs of SNPs in a block, and the average r^2 over all the SNP pairs, each of which belongs to two consecutive blocks. Overall, the Big-LD produces fewer blocks of larger size compared to other methods. The size discrepancy of LD blocks between the results of Big-LD and the other methods is larger for 1000G data than for HapMap data. The mean pairwise r^2 values within an LD block produced by Big-LD was slightly lower (0.472 for 1000G and 0.491 for HapMap) than the means for S-MIG++ (for 1000G), MIG++, and Haploview-CI, but higher than the means for MATILDE, S-MIG++ (for HapMap), Haploview-FGT, and Haploview-SS. Especially, the mean of pairwise r^2 values within an LD block produced by Big-LD is much higher than that of MATILDE (0.491 vs 0.285) for HapMap. On the other hand, average pairwise r^2 values across two consecutive LD blocks obtained by Big-LD was lowest (0.155 for 1000G and 0.141 for HapMap) compared to the other LD block construction methods except MATILDE. The distribution of pairwise r^2 values across two consecutive LD blocks obtained by Big-LD shows that more than 50% of the values are under 0.1 (Figure 3.6). The average pairwise D$'$ values within a LD block is lower than that of the other methods, moreover the average pairwise D$'$ values across consecutive blocks is much lower than that of the other methods (Table 3.8). The distribution of pairwise D$'$ values across two consecutive blocks excluding singleton blocks shows that the percentage of the values between 0.85 and 0.95 obtained from the results of Big-LD is much less than the percentage obtained from the other methods (Figure 3.7).

In Figure 3.8, distributions of block lengths are plotted for all LD block partition

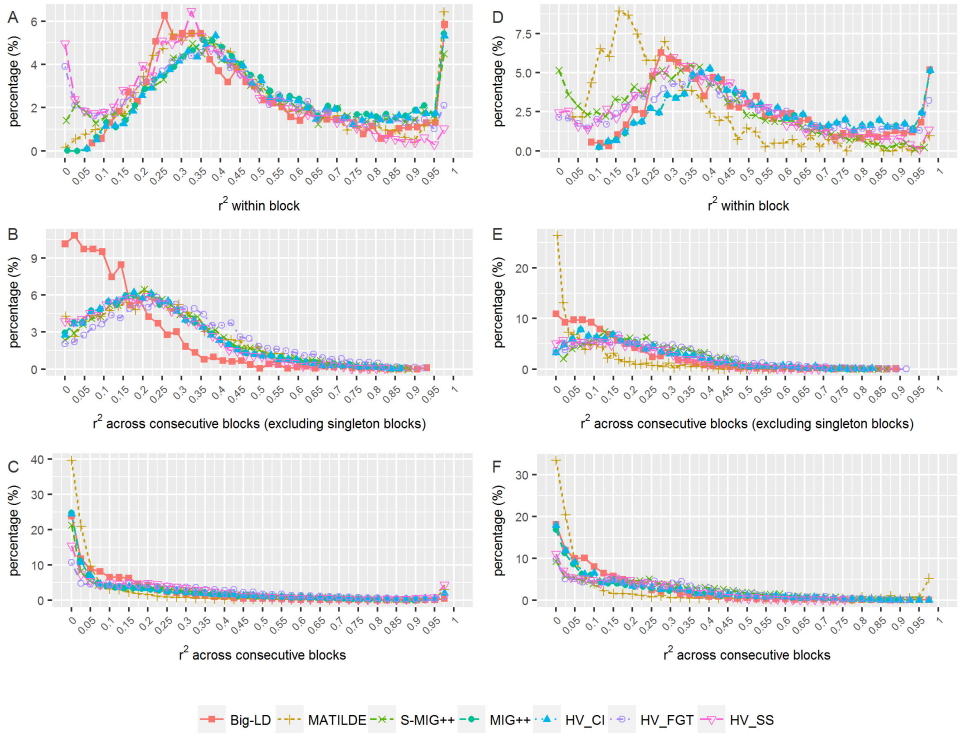


Figure 3.6: Distributions of the average pairwise r^2 between SNPs within a same LD block obtained by each method for chromosome 22 data of (A) 1000G and (B) HapMap datasets, and of the average pairwise r^2 between two SNPs in consecutive LD blocks excluding singleton blocks for (C) 1000G and (D) HapMap datasets and not excluding singleton blocks for (E) 1000G and (F) HapMap datasets.

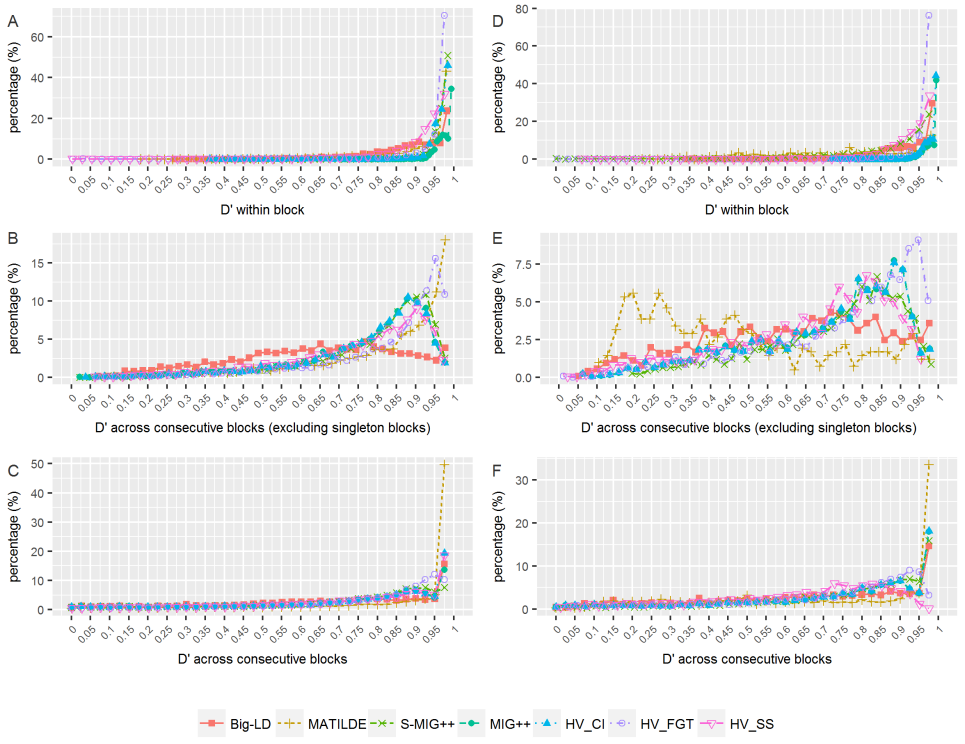


Figure 3.7: Distributions of the average pairwise D′ between SNPs within a same LD block obtained by each method for chromosome 22 data of (A) 1000G and (B) HapMap datasets, and of the average pairwise D′ between two SNPs in consecutive LD blocks excluding singleton blocks for (C) 1000G and (D) HapMap datasets and not excluding singleton blocks for (E) 1000G and (F) HapMap datasets.

Table 3.8: Summary of LD block partition results obtained for chromosome 22 region of the 1000G and HapMap datasets

	Method	N. of LD blocks		Size of LD blocks Mean (SD)		r^2 within block Mean(SD)	Average r^2 across consecutive blocks Mean (SD)		D′within block Mean (SD)	Average D′ across consecutive blocks Mean (SD)	
		All	Excluding singletons	Length (kb)	N. of SNPs		All	Excluding singletons		All	Excluding singletons
1000G	Big-LD	3654	2000	15427 (37001)	36.96 (69.23)	0.472 (0.243)	0.155 (0.156)	0.139 (0.128)	0.898 (0.099)	0.671 (0.278)	0.645 (0.215)
	MATILDE	13024	4361	6255 (9205)	15.34 (19.17)	0.466 (0.243)	0.12 (0.222)	0.263 (0.163)	0.919 (0.117)	0.823 (0.265)	0.814 (0.197)
	S-MIG++	11397	5253	5220 (10320)	13.22 (21.02)	0.474 (0.259)	0.217 (0.213)	0.269 (0.166)	0.977 (0.035)	0.711 (0.264)	0.797 (0.167)
	MIG++	11339	4658	5918 (13814)	14.79 (25.78)	0.517 (0.244)	0.181 (0.199)	0.252 (0.161)	0.978 (0.024)	0.711 (0.274)	0.781 (0.171)
	Haploview(CI)	11970	4418	6162 (13991)	15.4 (26.03)	0.513 (0.243)	0.214 (0.246)	0.249 (0.157)	0.976 (0.032)	0.733 (0.273)	0.782 (0.169)
	Haploview(FGT)	14160	7997	3298 (5619)	8.68 (8.95)	0.434 (0.254)	0.298 (0.237)	0.314 (0.185)	0.961 (0.099)	0.768 (0.24)	0.834 (0.169)
	Haploview(SS)	8811	5170	5665 (9345)	13.92 (17.66)	0.377 (0.22)	0.259 (0.26)	0.235 (0.148)	0.919 (0.124)	0.738 (0.258)	0.759 (0.184)
Hapmap	Big-LD	2018	1254	19520 (32894)	10.55 (12.85)	0.491 (0.232)	0.141 (0.142)	0.15 (0.127)	0.921 (0.089)	0.669 (0.273)	0.62 (0.23)
	MATILDE	1342	411	74364 (102641)	31.78 (32.99)	0.285 (0.18)	0.179 (0.288)	0.093 (0.106)	0.751 (0.19)	0.704 (0.305)	0.467 (0.245)
	S-MIG++	3192	1843	14316(31280)	6.86(8.25)	0.345 (0.21)	0.268 (0.204)	0.245 (0.14)	0.855 (0.188)	0.765 (0.232)	0.73 (0.179)
	MIG++	3644	1678	12718 (25251)	7.13 (8.52)	0.544 (0.233)	0.183 (0.183)	0.212 (0.149)	0.982 (0.023)	0.736 (0.26)	0.709 (0.203)
	Haploview(CI)	3643	1671	12871 (27410)	7.19 (8.87)	0.543 (0.233)	0.184 (0.184)	0.212 (0.149)	0.982 (0.023)	0.736 (0.26)	0.708 (0.203)
	Haploview(FGT)	3522	2238	9600 (15262)	5.68 (4.69)	0.449 (0.256)	0.258 (0.197)	0.256 (0.169)	0.977 (0.057)	0.726 (0.241)	0.748 (0.209)
	Haploview(SS)	2157	1714	14884 (26077)	7.91 (8.3)	0.395 (0.218)	0.184 (0.144)	0.192 (0.134)	0.935 (0.091)	0.644 (0.233)	0.667 (0.21)

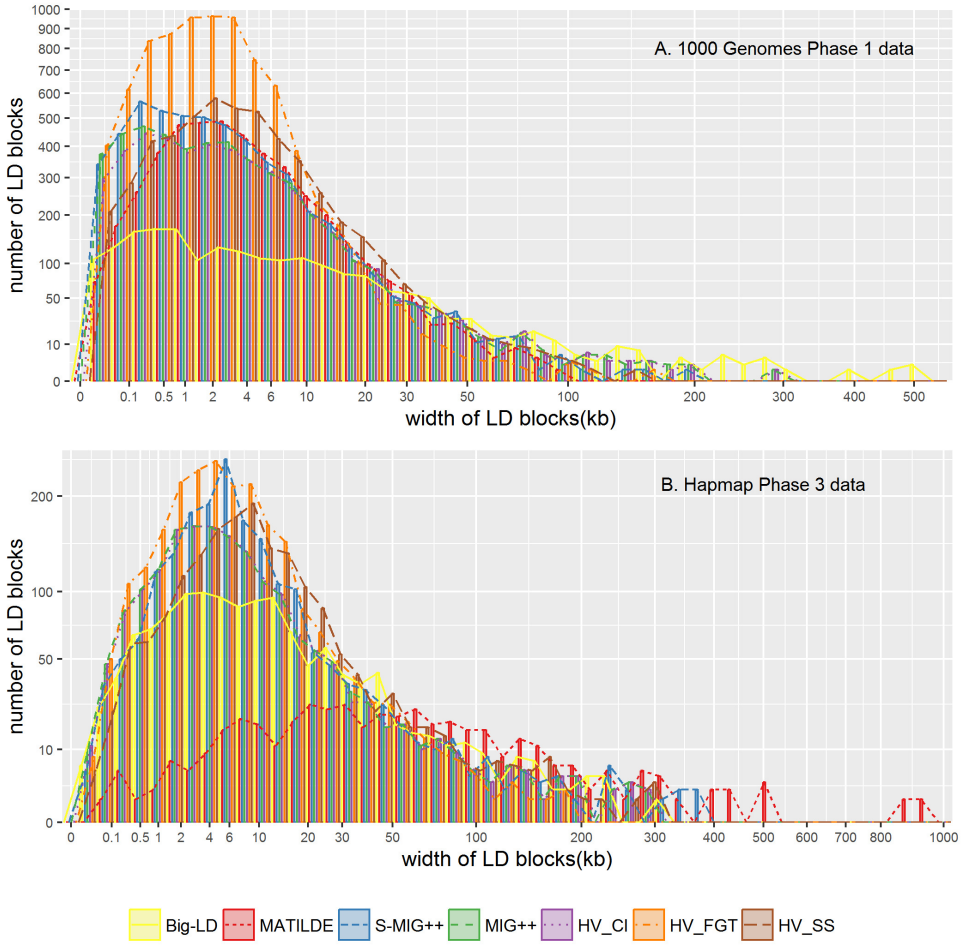


Figure 3.8: Distributions of the length of LD block obtained using chromosome 22 region of (A) 1000G dataset and (B) HapMap dataset. The length of the LD block is obtained as the difference of the bp position of starting and ending SNPs of a LD block.

Table 3.9: Distribution of LD block size (the number of SNPs in each block) obtained for chromosome 22 regions of the 1000G and HapMap datasets

		LDblock size											Sum	
	Method	1	2∼5	6∼10	11∼20	21∼30	31∼40	41∼50	51∼100	101∼200	201∼300	301∼	All	Excluding singletons
1000G	Big-LD	1654	794	232	213	154	92	77	241	139	28	30	3654	2000
	MATILDE	8663	1505	951	959	386	211	114	196	39	0	0	13024	4361
	S-MIG++	6144	2695	838	769	369	208	117	195	57	4	1	11397	5253
	MIG++	6681	2273	771	695	335	194	111	195	67	15	2	11339	4658
	Haploview(CI)	7552	2067	767	672	327	184	116	198	73	13	1	11970	4418
	Haploview(FGT)	6163	4004	1820	1454	469	160	49	39	2	0	0	14160	7997
	Haploview(SS)	3641	2138	1014	942	460	246	147	194	28	1	0	8811	5170
Hapmap	Big-LD	764	621	239	227	85	35	19	27	1	0	0	2018	1254
	MATILDE	931	49	56	81	73	46	32	56	16	2	0	1342	411
	S-MIG++	1349	1164	390	188	57	18	12	14	0	0	0	3192	1843
	MIG++	1966	1015	378	182	55	19	14	15	0	0	0	3644	1678
	Haploview(CI)	1972	1008	377	183	55	19	14	15	0	0	0	3643	1671
	Haploview(FGT)	1284	1431	552	210	37	5	3	0	0	0	0	3522	2238
	Haploview(SS)	443	862	498	252	55	29	5	13	0	0	0	2157	1714

methods. Table 3.9 reports distributions of the number of SNPs per block for all LD block partition methods. The LD blocks obtained from the 1000G dataset by Haploview-CI, MIG++, and S-MIG++ which are based on the LD block definition of Gabriel et al. [GSN$^+$02] show similar block size distributions. Compared to the other methods, Haploview-FGT produced more blocks of length up to about 10kb compared to other methods. Big-LD produces fewer blocks in size groups up until a certain size (20kb for 1000G and 15kb for HapMap) compared to the other LD block partition methods. MATILDE produces more LD blocks than Big-LD for 1000G, but produces fewer blocks than Big-LD for HapMap. Furthermore, especially for HapMap, Big-LD produces the most LD blocks of size at least 30 SNPs compared to the other methods except MATILDE.

In Figure 3.9, separate LD heatmaps for an example region on chromosome 22 show LD block boundaries obtained by all methods. This region exemplifies the case where a big LD block found by Big-LD is split into several small blocks by MATILDE, Haploview, MIG++, and S-MIG++. The big LD block between 23,250,737bp$\sim$23,644,677bp position found by Big-LD is split into 6$\sim$22 blocks by other methods for 1000G data and 2$\sim$7 blocks for HapMap data. The average r^2 values between the split consecutive blocks by the other methods are between 0.40$\sim$0.66 for 1000G data and between 0.54$\sim$0.65 for HapMap data (0.19 for MATILDE). In this example region, LD blocks found in 1000G data by Big-LD are also observed at similar locations in the HapMap data while there is greater discrepancy between 1000G and HapMap data results for the other LD block partition methods. We observed similarly that Big-LD produces large blocks which are usually split by the other methods. The LD blocks found by Big-LD in the two datasets also exhibit greater similarity compared to the other methods.

In Figure 3.10, we plot the distributions of LD block lengths obtained from

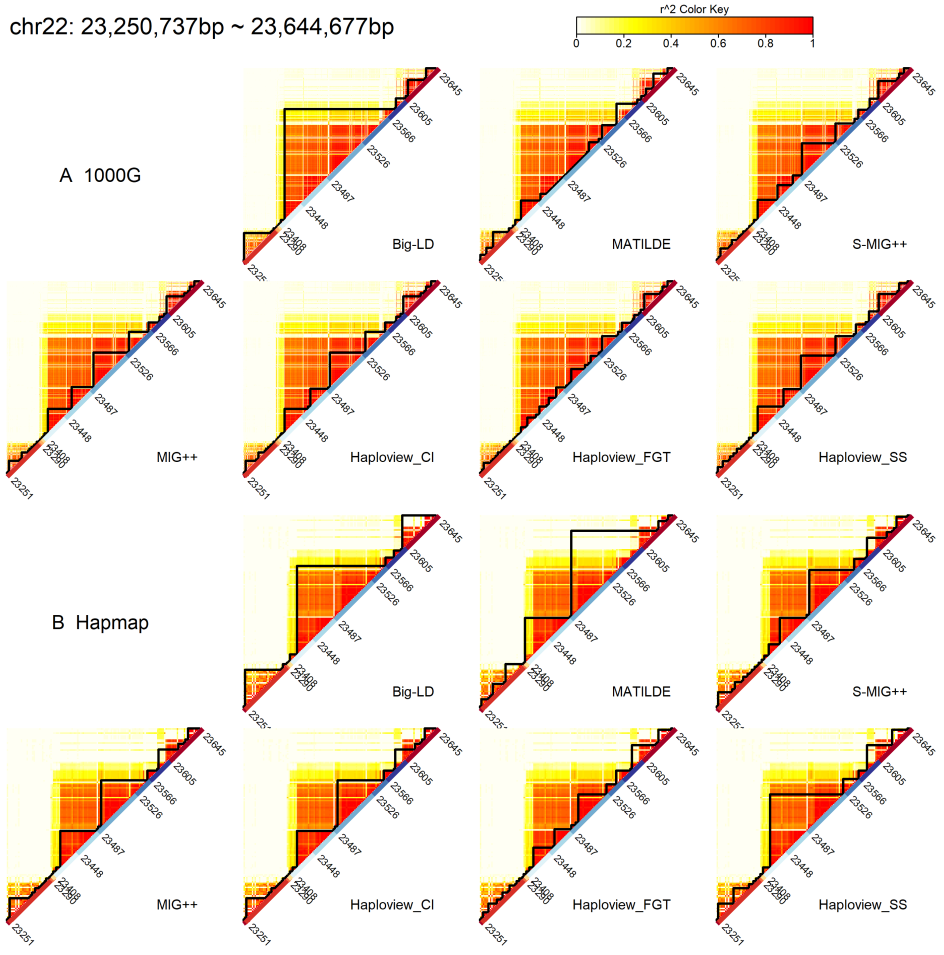


Figure 3.9: An example of LD heatmaps of the region chr22: 23,251kb ∼23,645kb with LD block partition results obtained from (A) 1000G dataset and (B) HapMap dataset. A solid triangle represents each LD block. Selected bp position in kb are shown.

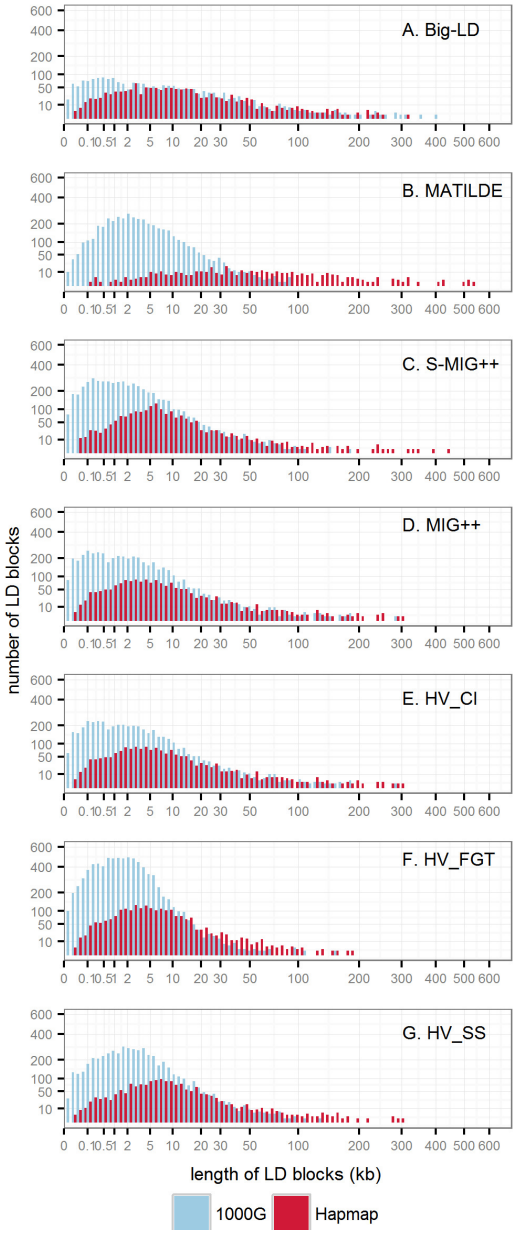


Figure 3.10: The distributions of the length of LD blocks obtained from 1000G and HapMap chromosome 22 datasets for each method of (A) Big-LD, (B) MATILDE, (C) S-MIG++, (D) MIG++, (E) Haloview (CI), (F) Haploview (FGT), and (G) Haploview (SS). The length of the LD block is obtained as the difference of the bp position of starting and ending SNPs of a LD block.

Table 3.10: Frequencies of common LD blocks obtained by different LD block partition in chromosome 22 regions of the 1000G and HapMap datasets

Method		# of LD blocks		Common block declaration criteria*			
		1000G	HapMap	90%	80%	70%	60%
Big-LD	All	3654	2018	172 (0.05 vs 0.09)	319 (0.09 vs 0.16)	441 (0.12 vs 0.22)	577 (0.16 vs 0.29)
	Excluding singletons	2000	1254	150 (0.08 vs 0.12)	297 (0.15 vs 0.24)	419 (0.21 vs 0.33)	555 (0.28 vs 0.44)
	Block width>=5kb	904	764	131 (0.14 vs 0.17)	247 (0.27 vs 0.32)	331 (0.37 vs 0.43)	432 (0.48 vs 0.57)
	Block width>=10kb	662	560	115 (0.17 vs 0.21)	197 (0.3 vs 0.35)	260 (0.39 vs 0.46)	340 (0.51 vs 0.61)
MATILDE	All	13024	1342	279 (0.02 vs 0.21)	295 (0.02 vs 0.22)	311 (0.02 vs 0.23)	334 (0.03 vs 0.25)
	Excluding singletons	4361	411	3 (0 vs 0.01)	19 (0 vs 0.05)	35 (0.01 vs 0.09)	58 (0.01 vs 0.14)
	Block width>=5kb	1569	382	3 (0 vs 0.01)	18 (0.01 vs 0.05)	28 (0.02 vs 0.07)	48 (0.03 vs 0.13)
	Block width>=10kb	816	343	2 (0 vs 0.01)	11 (0.01 vs 0.03)	19 (0.02 vs 0.06)	36 (0.04 vs 0.1)
S-MIG++	All	11397	3192	206 (0.02 vs 0.06)	342 (0.03 vs 0.11)	561 (0.05 vs 0.18)	799 (0.07 vs 0.25)
	Excluding singletons	5253	1843	56 (0.01 vs 0.03)	192 (0.04 vs 0.1)	411 (0.08 vs 0.22)	649 (0.12 vs 0.35)
	Block width>=5kb	1445	1055	46 (0.03 vs 0.04)	147 (0.1 vs 0.14)	296 (0.2 vs 0.28)	423 (0.29 vs 0.4)
	Block width>=10kb	743	604	35 (0.05 vs 0.06)	103 (0.14 vs 0.17)	189 (0.25 vs 0.31)	260 (0.35 vs 0.43)
MIG++	All	11339	3644	429 (0.04 vs 0.12)	650 (0.06 vs 0.18)	867 (0.08 vs 0.24)	1072 (0.09 vs 0.29)
	Excluding singletons	4658	1678	141 (0.03 vs 0.08)	362 (0.08 vs 0.22)	579 (0.12 vs 0.35)	784 (0.17 vs 0.47)
	Block width>=5kb	1355	816	95 (0.07 vs 0.12)	226 (0.17 vs 0.28)	343 (0.25 vs 0.42)	449 (0.33 vs 0.55)
	Block width>=10kb	707	499	53 (0.07 vs 0.11)	133 (0.19 vs 0.27)	201 (0.28 vs 0.4)	266 (0.38 vs 0.53)
Haploview(CI)	All	11970	3643	432 (0.04 vs 0.12)	660 (0.06 vs 0.18)	870 (0.07 vs 0.24)	1078 (0.09 vs 0.3)
	Excluding singletons	4418	1671	145 (0.03 vs 0.09)	373 (0.08 vs 0.22)	583 (0.13 vs 0.35)	791 (0.18 vs 0.47)
	Block width>=5kb	1334	809	98 (0.07 vs 0.12)	234 (0.18 vs 0.29)	344 (0.26 vs 0.43)	456 (0.34 vs 0.56)
	Block width>=10kb	704	493	57 (0.08 vs 0.12)	141 (0.2 vs 0.29)	203 (0.29 vs 0.41)	273 (0.39 vs 0.55)
Haploview(FGT)	All	14160	3522	258 (0.02 vs 0.07)	466 (0.03 vs 0.13)	701 (0.05 vs 0.2)	977 (0.07 vs 0.28)
	Excluding singletons	7997	2238	130 (0.02 vs 0.06)	338 (0.04 vs 0.15)	573 (0.07 vs 0.26)	849 (0.11 vs 0.38)
	Block width>=5kb	1622	1077	55 (0.03 vs 0.05)	132 (0.08 vs 0.12)	219 (0.14 vs 0.2)	316 (0.19 vs 0.29)
	Block width>=10kb	558	615	28 (0.05 vs 0.05)	59 (0.11 vs 0.1)	98 (0.18 vs 0.16)	133 (0.24 vs 0.22)
Haploview(SS)	All	8811	2157	151 (0.02 vs 0.07)	290 (0.03 vs 0.13)	488 (0.06 vs 0.23)	688 (0.08 vs 0.32)
	Excluding singletons	5170	1714	106 (0.02 vs 0.06)	245 (0.05 vs 0.14)	443 (0.09 vs 0.26)	643 (0.12 vs 0.38)
	Block width>=5kb	1654	1062	67 (0.04 vs 0.06)	146 (0.09 vs 0.14)	273 (0.17 vs 0.26)	385 (0.23 vs 0.36)
	Block width>=10kb	817	665	42 (0.05 vs 0.06)	84 (0.1 vs 0.13)	141 (0.17 vs 0.21)	206 (0.25 vs 0.31)

* A (B vs C): A is the number of common blocks according to the common block declaration criteria, B is the relative frequency of the blocks found in the 1000 Genomes Project Data and the C is the relative frequency of the blocks found in the HapMap data. An LD block is declared as a common block found in two datasets if the relative percentage length of common area of two LD block results over the length of the entire LD block (in both results) is greater than the criterion.

Table 3.11: Average of haplotype diversity indices of LD blocks obtained for chromosome 22 regions of the 1000G and HapMap datasets by different LD block partition methods

Method	Haplotype diversity index	
	1000G Mean (SD)	HapMap Mean (SD)
Big-LD	0.911 (0.157)	0.985 (0.023)
MATILDE	0.969 (0.076)	0.870 (0.192)
S-MIG++	0.987 (0.026)	0.994 (0.01)
MIG++	0.983 (0.037)	0.993 (0.01)
HV_CI	0.982 (0.039)	0.993 (0.01)
HV_FGT	0.995 (0.008)	0.996 (0.005)
HV_SS	0.983 (0.028)	0.992 (0.012)

1000G and HapMap data together for each LD block partition method. In Big-LD, the number of small blocks of up to 2kb is usually larger in 1000G data compared to HapMap data while the numbers of blocks greater than 2kb are usually similar in both datasets. Other LD block partition methods produces more blocks of up to 20 kb from 1000G data than from HapMap data, suggesting the differences in block partition results for the datasets with different density is greater in the other methods than in Big-LD. We also examined actual overlap proportions of LD blocks obtained from both datasets for each method (Table 3.10). The proportion of overlapping LD blocks in each dataset over the number of all LD blocks produced in each dataset is larger for Big-LD compared to other methods. This tendency is even stronger when comparing only the larger blocks. For example, when we use the overlap declaration criterion of 80% and limit the comparison to the blocks greater than 10kb, the overlap proportions of LD blocks produced by Big-LD are 30% of the number of such blocks in 1000G data and 35% of the number of blocks in HapMap data while these overlap proportions are 1$\sim$29% for the other methods.

We also evaluated haplotype diversity measured by the haplotype diversity index for a block [PBH$^+$01, ZDC$^+$02]. The haplotype diversity index is defined as

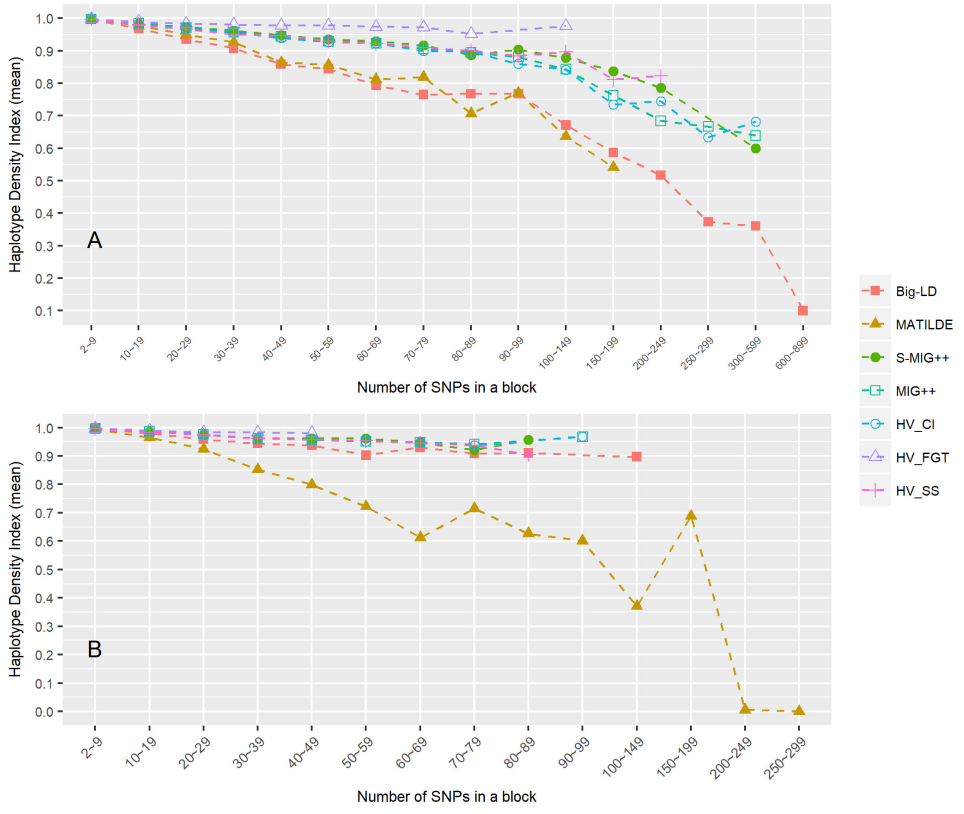


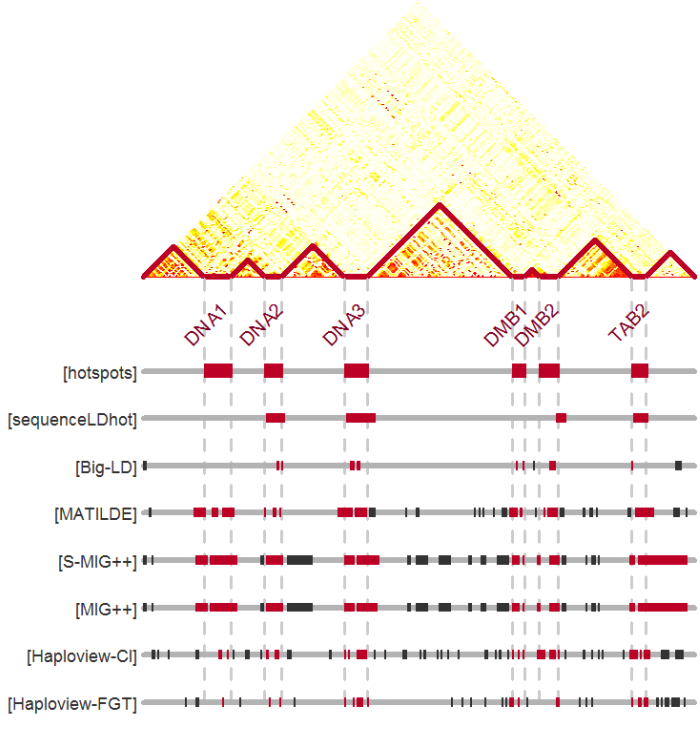
Figure 3.11: Distribution of haplotype diversity index of LD blocks obtained by different LD block partition methods for the chromosome 22 regions of (A) 1000G and (B) HapMap datasets.

the ratio of the number of common haplotypes in the block over the total number of haplotypes within the block. The mean haplotype diversity indices of the blocks obtained by Big-LD for 1000G and HapMap are 0.911 and 0.985, respectively, which are lower compared to those obtained by the other methods except MATILDE (0.87 for HapMap) (Table 3.11). The distribution of haplotype diversity indices shows that the haplotype diversity of the blocks obtained by Big-LD decreases as the LD block size increases (Figure 3.11).

3.3.4 LD block and recombination hotspots

For the MHC dataset of 50 north-European British semen donors, we obtained the LD block partition results using Big-LD, MATILDE, three Haploview methods (CI, FGT, SS), MIG++ and S-MIG++ to compare with the true recombination hotspot locations obtained by sperm-typing experiment on the same sample of individuals in the MHC dataset as reported in [JKN01]. For this comparison, we regarded the region between non-singleton LD blocks as the block boundary regions. Also, we obtained estimated hotspots from MHC data using sequenceLDhot [Fea06] to compare with the true recombination hotspot regions.

For sequenceLDhot, we set the criteria to declare hotspots when the log-likelihood ratio over the background recombination rate is greater than 12. Also, we set the sliding window option for sequenceLDhot as 2kb windows moving about every 1kb point. In Figure 3.12, the LD heatmap of this region and the block boundary locations or hotspot locations found by all methods are shown along with the true discovery rate (TDR) defined as the ratio of the number of block boundaries or hotspots coinciding with the true hotspots over the number of all true hotspots found by the sperm-typing experiment (total of 6 hotspots). The false discovery rate (FDR) is defined as the ratio of the number of block boundaries or hotspots not in the true hotspot regions over the number of all block boundaries or hotspots found by the given method. Big-LD finds five out of six true hotspots (TDR of 83.3%) and shows low FDR (27.3%). The other LD block partition methods including MATILDE, three Haploview methods, and MIG++ yield 100% TDR but also shows high FDRs greater than 50%. The sequenceLDhot yields lower TDR (66.7%) than the Big-LD, but shows zero FDR. We also compared LD block partition results of Big-LD and other methods applied to 1000G and HapMap data with



	TDR	FDR
[hotspots]	4/6(66.7%)	0/4(0%)
[sequenceLDhot]	5/6(83.3%)	3/11(27.3%)
[Big-LD]	6/6(100%)	18/31(58.1%)
[MATILDE]	6/6(100%)	14/25(56%)
[S-MIG++]	6/6(100%)	14/25(56%)
[MIG++]	6/6(100%)	30/46(65.2%)
[Haploview-CI]	6/6(100%)	19/32(59.4%)
[Haploview-FGT]		

Figure 3.12: The LD heatmap of the MHC dataset with true recombination hotspot positions found by sperm-typing experiments in [JKN01], LD block boundary locations found by Big-LD, MATILDE, Haploview (CI, FGT, SS), MIG++ and hotspot locations found by sequenceLDhot. The red bars show LD block boundaries overlapping with true recombination hotspots and the black bars show LD block boundaries not overlapping with them. TDR is the number of the true recombination hotspots overlapping with the block boundaries found by each method over the number of true recombination hotspots which is 6, and FDR is the number of non-overlapping block boundaries with the true recombination hotspots among the ones found by each method over the all block boundaries found by each method.

Table 3.12: True and false discovery rates of estimated recombination hotspots obtained from sequenceLDhot by different LD block partition methods

	1000G		HapMap	
	TDR*	FDR**	TDR*	FDR**
Big-LD	0.655 (468/714)	0.652 (1305/2000)	0.858 (296/345)	0.734 (921/1255)
MATILDE	0.675 (482/714)	0.858 (3744/4362)	0.406 (140/345)	0.665 (274/412)
S-MIG++	0.993 (709/714)	0.773 (4059/5253)	0.884 (305/345)	0.812 (1497/1843)
MIG++	0.978 (698/714)	0.760 (3540/4659)	0.980 (338/345)	0.780 (1309/1679)
Haploview(CI)	0.971 (693/714)	0.764 (3376/4419)	0.980 (338/345)	0.779 (1302/1672)
Haploview(FGT)	0.985 (703/714)	0.837 (6692/7997)	0.991 (342/345)	0.821 (1838/2239)
Haploview(SS)	0.916 (654/714)	0.780 (4035/5170)	0.928 (320/345)	0.779 (1335/1714)

* C (B / A): A is the number of all hotspots found by sequenceLDhot. B is the number of LD block boundaries obtained by the given method coinciding with the hotspots. C = B / A.

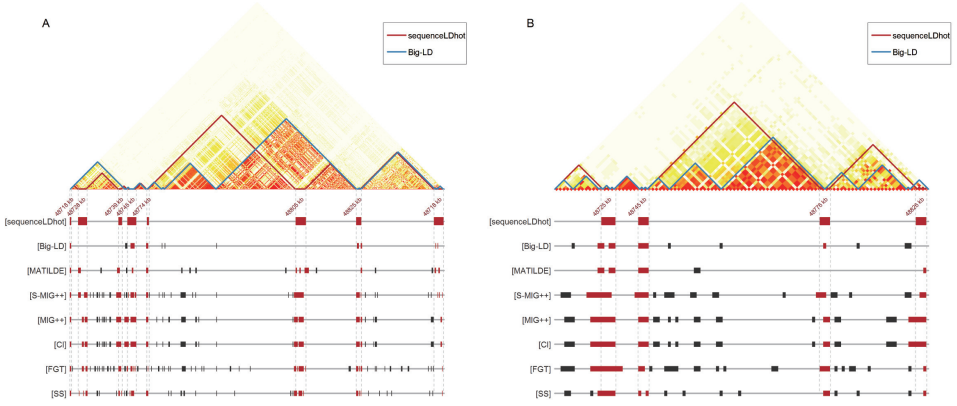
** F (E / D): D is the number of the LD block boundaries obtained by the given method. E is the number of the LD block boundaries not coinciding with the hotspots. F = E / D.

the recombination hotspot locations obtained for these datasets by sequenceLDhot to see if the LD block boundaries and estimated recombination hotspot regions coincide with each other. Considering the recombination hotspot regions estimated by the sequenceLDhot as the true hotspot regions, we obtained TDR and FDR similarly defined for the MHC dataset analysis (Table 3.12). The TDR for Big-LD results (65.5% in 1000G and 85.8% in HapMap) is lowest compared to the TDR for the other methods (67%$\sim$99%) except MATILDE (40% in HapMap) since other methods produced smaller blocks with more chance to coincide with the estimated recombination hotspot regions. However, the FDR for Big-LD (65.2% for 1000G and 73.4% for HapMap) is lowest among all LD block partition methods except MATILDE (66.5% in HapMap). On average, the distance between consecutive estimated recombination hotspots is 46,699 kb (99.5 SNPs) for 1000G data and 94,802 kb (38.5 SNPs) for HapMap data, which are twice and four times the distance between consecutive block boundary positions obtained by Big-LD for 1000G and HapMap data respectively. LD heatmaps of some example regions compare the

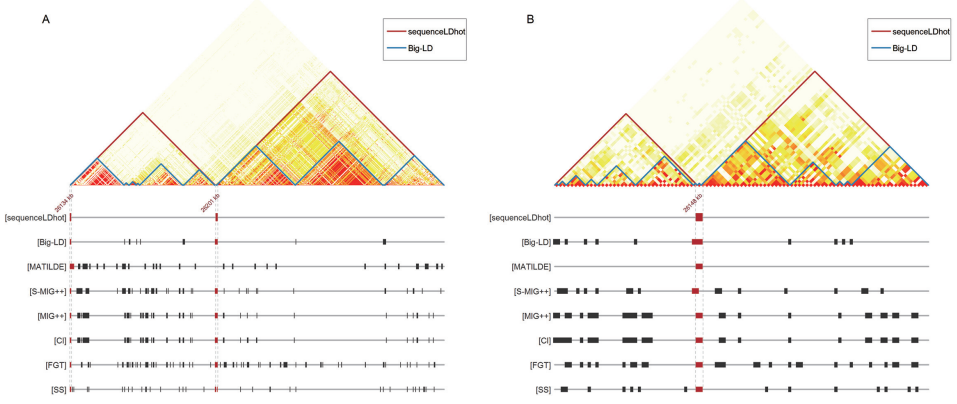
block boundaries of all partition methods and estimated recombination hotspot positions obtained by the sequenceLDhot (Figure 3.13).

3.3.5 Multi-SNP association experiments using the results of different block partition methods

We performed power analysis using simulated data based on part of 1000 Genomes Project dataset under multi-SNP association models using 7 regions of lung cancer related genes (CRKL, MIF, GSTT1, ZNRF3, PATZ1, TIMP3, LGALS2). We selected a big LD block which overlaps each gene region among the blocks produced by Big-LD. Then we set a 10 causal SNP trait model by randomly selecting 10 causal SNPs within this block region. The generation model for phenotype Y is set additively such as $Y = \sum_{i=1}^{10} \beta_i X_i + \varepsilon$ where X_i is the genotype (count of minor alleles) of the i^{th} causal SNP, β_i is the additive effect, and ε is the error with variance σ^2. We choose σ^2 values for each model to make the power values be not too low or high and set $\beta_i = 1$ for all causal SNPs. We performed two experiments for each region using two type I error threshold; 1) region-wide Bonferroni correction (0.05 divided by the number of blocks in the test region) and 2) chromosome-wide Bonferroni correction (0.05 divided by the number of blocks in the entire chromosome 22) [McD09]. We simulated $N = 1000$ datasets under each generation model. We applied seven multi-SNP analysis methods to each simulated dataset and obtained the p-values of each statistic. Seven methods include multi-SNP regression based tests: Wald (quadratic test), LC-B and LC-Z (linear combination tests), MLC-B and MLC-Z (quadratic test with dimension reduction) (see [YSP$^+$17] for detailed definition), and the marginal SNP score based tests such as SKAT [BCZ10, KLL$^+$08] and SKAT-O [LEB$^+$12, WLC$^+$11]. Using the significance level which is Bonferroni corrected for the number of blocks under two scenarios (chromosome-wide



(1) chr22:48,718,282bp∼48,836,425bp



(2) chr22: 25,975,254bp∼26,235,629bp

Figure 3.13: LD heatmaps of some example regions with LD block boundary marks found by Big-LD and other LD block partition methods along with the estimated recombination hotspots regions obtained by sequenceLDhot for (A)1000G dataset and (B) HapMap dataset

analysis and region-wide analysis), the power of each statistic for each block partition result has been empirically obtained.

The power results are given in Table 3.13 (region-wide results) and Table 3.14 (chromosome-wide results). We observed that the power of Wald, MLC-B, and MLC-Z are usually lower when multiple SNP effects are combined over a big LD block produced by Big-LD. The power of LC-B, LC-Z, SKAT and SKAT-O obtained for the block produce by Big-LD was usually higher than the power based on other LD block partition methods. The power of the quadratic test Wald and MLC seemed to be diluted more for Big-LD due to the increased degrees of freedom when including more SNPs in a block. These results are also consistent with the results reported in [DLS14].

Table 3.13: Empirical power of multi-SNP association tests corresponding to adjusted region-wide significance level by Bonferroni method.

Region	Method	N. of blocks	Wald	LC-B	LC-Z	MLC-B	MLC-Z	SKAT	SKAT-O
CRKL	Big-LD	1	0.393	0.828	0.832	0.346	0.344	0.845	0.828
21,017,148∼213,153,84	S-MIG++	82	0.236	0.314	0.312	0.273	0.273	0.309	0.29
(747 SNPs)	MIG++	89	0.304	0.304	0.304	0.304	0.304	0.304	0.304
σ =10	Haploview(CI)	148	0.266	0.266	0.266	0.266	0.266	0.266	0.266
	Haploview(FGT)	129	0.263	0.284	0.277	0.263	0.263	0.293	0.263
	Haploview(SS)	60	0.23	0.342	0.331	0.264	0.263	0.362	0.321
MIF	Big-LD	1	0.427	0.509	0.484	0.701	0.695	0.674	0.764
24,211,980∼24,238,079	S-MIG++	5	0.355	0.455	0.446	0.495	0.485	0.488	0.573
(67 SNPs)	MIG++	3	0.262	0.519	0.514	0.523	0.537	0.553	0.609
σ =10	Haploview(CI)	3	0.262	0.519	0.514	0.523	0.537	0.553	0.609
	Haploview(FGT)	8	0.353	0.486	0.447	0.478	0.484	0.441	0.503
	Haploview(SS)	2	0.399	0.585	0.476	0.665	0.68	0.615	0.659
GSTT1	Big-LD	1	0.386	0.92	0.893	0.792	0.794	0.923	0.924
24,344,926∼24,385,697	S-MIG++	20	0.602	0.618	0.62	0.618	0.62	0.605	0.605
(36 SNPs)	MIG++	15	0.593	0.658	0.651	0.658	0.651	0.638	0.637
σ =25	Haploview(CI)	36	0.552	0.552	0.552	0.552	0.552	0.552	0.552
	Haploview(FGT)	36	0.552	0.552	0.552	0.552	0.552	0.552	0.552
	Haploview(SS)	17	0.553	0.649	0.615	0.553	0.553	0.635	0.627
ZNRF3	Big-LD	1	0.564	0.936	0.952	0.829	0.825	0.95	0.944
29,523,628∼29,556,739	S-MIG++	10	0.72	0.754	0.791	0.667	0.667	0.778	0.793
(44 SNPs)	MIG++	11	0.719	0.82	0.82	0.82	0.82	0.808	0.807
σ =14	Haploview(CI)	11	0.719	0.82	0.82	0.82	0.82	0.808	0.807

	Haploview(FGT)	10	0.679	0.729	0.659	0.754	0.731	0.816	0.761
	Haploview(SS)	6	0.73	0.757	0.803	0.73	0.73	0.862	0.826
PATZ1	Big-LD	1	0.29	0.934	0.934	0.543	0.55	0.879	0.897
31,535,995∼31,787,234	S-MIG++	42	0.625	0.633	0.637	0.625	0.625	0.625	0.625
(370 SNPs)	MIG++	27	0.67	0.67	0.67	0.67	0.67	0.67	0.67
σ =15	Haploview(CI)	23	0.686	0.686	0.686	0.686	0.686	0.686	0.686
	Haploview(FGT)	110	0.52	0.52	0.52	0.52	0.52	0.52	0.52
	Haploview(SS)	69	0.575	0.575	0.575	0.575	0.575	0.575	0.575
TIMP3	Big-LD	1	0.254	0.854	0.858	0.464	0.461	0.865	0.842
33,148,102∼33,237,191	S-MIG++	21	0.416	0.498	0.502	0.416	0.416	0.497	0.47
(222 SNPs)	MIG++	19	0.504	0.504	0.508	0.504	0.504	0.504	0.504
σ =15	Haploview(CI)	19	0.504	0.504	0.508	0.504	0.504	0.504	0.504
	Haploview(FGT)	31	0.3	0.439	0.443	0.38	0.382	0.431	0.41
	Haploview(SS)	15	0.39	0.525	0.523	0.39	0.39	0.537	0.485
LGALS1	Big-LD	1	0.255	0.864	0.85	0.523	0.519	0.839	0.836
38,008,395∼38,077,250	S-MIG++	12	0.254	0.526	0.545	0.441	0.44	0.5	0.52
(117 SNPs)	MIG++	9	0.241	0.564	0.542	0.315	0.318	0.525	0.542
σ =20	Haploview(CI)	9	0.241	0.564	0.542	0.315	0.318	0.525	0.542
	Haploview(FGT)	20	0.421	0.462	0.495	0.421	0.421	0.421	0.449
	Haploview(SS)	9	0.241	0.524	0.492	0.324	0.322	0.494	0.515

* The significance level adjusted for multiple testing is obtained by 0.05/(N. of blocks)

Table 3.14: Empirical power of multi-SNP association tests corresponding to adjusted chromosome-wide significance level by Bonferroni method.

Region	Method	N. of blocks	Wald	LC-B	LC-Z	MLC-B	MLC-Z	SKAT	SKAT-O
CRKL	Big-LD	3101	0.248	0.928	0.927	0.347	0.335	0.94	0.908
21,017,148∼213,153,84	S-MIG++	11397	0.8	0.907	0.905	0.866	0.864	0.862	0.858
(747 SNPs)	MIG++	11339	0.902	0.902	0.902	0.902	0.902	0.902	0.902
σ =5	Haploview(CI)	11970	0.901	0.901	0.904	0.901	0.901	0.901	0.901
	Haploview(FGT)	14160	0.88	0.905	0.893	0.887	0.887	0.908	0.88
	Haploview(SS)	8811	0.825	0.915	0.905	0.863	0.857	0.908	0.891
MIF	Big-LD	3101	0.573	0.375	0.345	0.921	0.912	0.828	0.886
24,211,980∼24,238,079	S-MIG++	11397	0.672	0.62	0.594	0.831	0.853	0.718	0.803
(67 SNPs)	MIG++	11339	0.47	0.621	0.595	0.831	0.854	0.711	0.768
σ =5	Haploview(CI)	11970	0.467	0.616	0.591	0.831	0.849	0.711	0.764
	Haploview(FGT)	14160	0.779	0.761	0.687	0.895	0.894	0.723	0.788
	Haploview(SS)	8811	0.657	0.602	0.396	0.918	0.925	0.727	0.746
GSTT1	Big-LD	3101	0.185	0.913	0.862	0.768	0.777	0.908	0.898
24,344,926∼24,385,697	S-MIG++	11397	0.803	0.826	0.829	0.826	0.829	0.803	0.803
(36 SNPs)	MIG++	11339	0.726	0.83	0.831	0.83	0.831	0.766	0.781
σ =15	Haploview(CI)	11970	0.827	0.827	0.827	0.827	0.827	0.827	0.827
	Haploview(FGT)	14160	0.818	0.818	0.818	0.818	0.818	0.818	0.818
	Haploview(SS)	8811	0.701	0.856	0.814	0.701	0.701	0.811	0.81
ZNRF3	Big-LD	3101	0.386	0.823	0.862	0.765	0.769	0.88	0.869
29,523,628∼29,556,739	S-MIG++	11397	0.765	0.782	0.808	0.744	0.732	0.794	0.813
(44 SNPs)	MIG++	11339	0.766	0.836	0.827	0.836	0.827	0.794	0.813

σ =9	Haploview(CI)	11970	0.762	0.834	0.823	0.834	0.823	0.789	0.81
	Haploview(FGT)	14160	0.729	0.685	0.592	0.788	0.786	0.778	0.687
	Haploview(SS)	8811	0.685	0.71	0.782	0.767	0.758	0.838	0.78
PATZ1	Big-LD	3101	0.114	0.882	0.89	0.481	0.481	0.884	0.86
31,535,995∼31,787,234	S-MIG++	11397	0.801	0.83	0.834	0.801	0.801	0.79	0.778
(370 SNPs)	MIG++	11339	0.845	0.845	0.845	0.845	0.845	0.845	0.845
σ =10	Haploview(CI)	11970	0.844	0.844	0.844	0.844	0.844	0.844	0.844
	Haploview(FGT)	14160	0.737	0.821	0.831	0.737	0.737	0.771	0.742
	Haploview(SS)	8811	0.765	0.847	0.854	0.765	0.765	0.808	0.775
TIMP3	Big-LD	3101	0.114	0.882	0.89	0.481	0.481	0.884	0.86
33,148,102∼33,237,191	S-MIG++	11397	0.801	0.83	0.834	0.801	0.801	0.79	0.778
(222 SNPs)	MIG++	11339	0.845	0.845	0.845	0.845	0.845	0.845	0.845
σ =8	Haploview(CI)	11970	0.844	0.844	0.844	0.844	0.844	0.844	0.844
	Haploview(FGT)	14160	0.737	0.821	0.831	0.737	0.737	0.771	0.742
	Haploview(SS)	8811	0.765	0.847	0.854	0.765	0.765	0.808	0.775
LGALS1	Big-LD	3101	0.184	0.953	0.939	0.698	0.703	0.903	0.9
38,008,395∼38,077,250	S-MIG++	11397	0.718	0.915	0.92	0.873	0.875	0.887	0.888
(117 SNPs)	MIG++	11339	0.381	0.911	0.897	0.752	0.743	0.885	0.892
σ =10	Haploview(CI)	11970	0.377	0.908	0.896	0.748	0.742	0.885	0.889
	Haploview(FGT)	14160	0.843	0.895	0.907	0.843	0.843	0.843	0.877
	Haploview(SS)	8811	0.455	0.888	0.85	0.779	0.783	0.87	0.874

* The significance level adjusted for multiple testing is obtained by 0.05/(N. of blocks)

3.4 Discussion

Big-LD occupies much more memory (at most 3042Mb for 1000G chromosome 22 data) compared to S-MIG++ (less than 25Mb for for 1000G chromosome 22 data) mostly due to the clique finding algorithms in R *igraph* package. However, Big-LD can run without problems on high performance computers with more than 4GB memory.

The LD block partition methods adopting a narrow sense definition of LD blocks tend to separate SNPs that are in strong LD if they have some SNPs in between that are in low LD with those "strong LD" SNPs. This case is observed more often when more SNPs are genotyped in the same region such as with sequencing data. As a result, the existing LD block construction algorithms usually split large LD block into small blocks as evident in the application to the 1000G dataset. For Big-LD, the problem of "interruption by low LD SNPs" is less likely to affect the LD block construction results because Big-LD uses an agglomerative approach that starts by identifying small communities of SNPs, i.e. the SNPs in each LD bin region (from starting SNP to ending SNP), and proceeds by merging these communities. By including all SNPs that are located between starting and ending SNPs of each LD bin in the initial communities of SNPs, low LD SNPs in the middle of a high LD region do not severely affect the block partition results. Allowing this region of low LD between strongly associated SNP pairs goes against the criteria for evaluation of haplotype block construction results suggested by Wall and Pritchard [WP03]: "absence of holes" (If a pair of SNPs are in strong LD, then both SNPs should be in strong LD with SNPs that lie in between). When the purpose of haplotype block construction is to find tag SNPs, the presence of holes in the block region can yield cases where tag SNPs are not capturing the effects of some SNPs

in the block. When, however, the purpose of haplotype block construction is to find blocks with low between-block correlation or blocks revealing population genetic structure such as recombination hotspots, especially with high density sequencing data, the "absence of holes" principle may not serve the purpose.

We found that SNPs in the LD blocks produced by Big-LD show higher r^2 and D' values when they are in the same block and lower r^2 and D' values when they are in different blocks. In the three Haploview methods, as well as in MIG++, S-MIG++, and MATILDE, SNPs in the same block have similar, slightly higher or lower r^2 and D' values compared to Big-LD, but SNPs in consecutive blocks also have higher r^2 and D' values. This result shows that Big-LD produces more optimized blocks compared to the other methods in the sense that r^2 values are high within the same block and low between different blocks. In particular, if we want to use LD block construction results to choose SNP sets for genetic association analysis using multi-SNP methods where the effects of multiple SNPs are combined, low r^2 values between different SNP sets would result in less correlated null statistics corresponding to multiple hypotheses. To adjust p-values of the association analysis results by the genomic significance level using Bonferroni method, the independence of null hypotheses representing many multi-SNP statistics is needed [McD09]. In our evaluation of the power of various multi-SNP association tests based on various LD block construction results, linear combination type or marginal SNP score based tests usually show better power for big LD block region produced by Big-LD under multiple causal SNP models, whereas the power of quadratic sum type tests was lower when the effects of multiple SNPs are quadratically combined over a large block.

When we compared the LD block partition results obtained using 1000G and HapMap datasets, we observed that the LD blocks found by Big-LD from two data-

sets overlap more than the other methods, especially for large size blocks. This result suggests that Big-LD finds more invariant, less data-dependent big LD blocks regardless of the marker-density of the data. Also, the difference between the LD block construction results of Big-LD and the other methods was even greater in 1000G data than in HapMap data. With the dense SNPs in the 1000G data, CLQ-D usually produces a block LD structure with more LD bins that consist of non-consecutive SNPs compared to HapMap data (Figure 3.9). Since the ranges of LD bins are more invariant to the marker-density of genotyped SNPs, Big-LD produces more invariant big LD blocks by the "bottom-up" type clustering of LD bins. For the other methods such as MATILDE, Haploview, S-MIG++ or MIG++, the block partition results are very sensitive to the SNP marker-density and/or the selection of SNPs.

How the locations of recombination hotspots are related to LD blocks is of major interest since a recombination hotspot is the one of the main reasons for LD block formation. In early days, without clear operational definitions for LD block, researchers found by observing some regions that true recombination hotspot locations confirmed by sperm-typing experiments coincide with LD block boundaries found as LD break points [JKN01]. After several LD block definitions were introduced, other researchers found that LD blocks defined in a narrow sense can be formed without recombination hotspots [WAZ$^+$02]. In our comparisons of LD block partition results obtained by Big-LD algorithm and the other methods with the experimentally determined recombination hotspots for the same MHC dataset used in [JKN01], we found that LD blocks found by Big-LD agree better with the true recombination hotspots than the existing methods. We also saw better agreement between the LD block locations found by Big-LD and the estimated hotspot regions obtained by the sequenceLDhot compared to the other LD block partition

methods. However, the agreement proportions between the results of Big-LD and sequenceLDhot do not exceed 50% even with relaxed criterion such as 60% overlap to be declared as a shared LD block. Since the estimated recombination hotspots by sequenceLDhot for MHC data also did not coincide with the real hotspots perfectly (lower TDR than Big-LD), more comprehensive data with accurate information on hotspots for large regions would be helpful to further determine the relationship between the LD blocks found by Big-LD and recombination hotspots.

Most graph clustering algorithms require presetting the number of clusters before applying clustering steps [dAH15]. Big-LD does not require prior specification of the number of clusters. Instead, the clusters are chosen at each step of the greedy process which picks out a set of independent cliques of the interval graph that do not overlap each other. It uses an algorithm to choose the maximum weight independent set of a graph where the vertices are all cliques in the interval graph and the edges represent the overlap between the intervals covered by each clique. By applying this process until the remaining genomic regions are all covered by non-overlapping intervals, i.e. the LD blocks, Big-LD can automatically determine how many clusters should be present in the final partitioning results.

In summary, the new LD block construction method Big-LD produces larger LD blocks than existing methods since it adopts a wide sense definition for LD block, which allows a "hole" of low LD SNPs to be present between strongly associated SNP pairs. Big-LD determines the number of blocks using a maximum weight independent set selection algorithm. The LD blocks produced by Big-LD agree better than existing methods with the recombination hotspot locations determined by sperm-typing experiments. The observed average runtime of Big-LD for 13,288,240 non-monomorphic SNPs from 1000 Genomes Project autosome data (286 East Asians) is about 5.83 hours, which is a significant improvement over the

existing methods.

Chapter 4

Comparisons of linkage disequilibrium blocks of different populations for positive selection

4.1 Background

Linkage disequilibrium (LD) is a dependency between alleles at different loci. It is known that the strength of LD between two loci decreases as the genetic distance between them increases [PP01, Mor05]. However, actual LD patterns revealed from the analysis of human genetic data show mosaic block-like structures [JKN01, WAZ+02]. The underlying causes of LD block structure have been attributed to population genetic phenomena such as mutation, selection, recombination or genetic drift [MMH+04, Sla08]. Especially, recombination hotspots where the recombination rates arise rapidly compared to the background rates have been suspected to be one of the major reasons of block formation [Fea06, Mor05].

The distribution of LD over a region has been looked into in relation to positive selection. The existence of long ranging haplotype (LRH) around an allele has been suggested as an evidence of positive selection [SRH+02, SVF+07, VKWP06]. The frequency of an allele with higher fitness might have increased rapidly, which yields a strong extended LD over surrounding variants [SRH+02]. Several statistical tests have been developed such as LRH [SRH+02], iHS [VKWP06], and XP-EHH [SVF+07] based on this assumption of the relationship between long range haplo-

type and positive selection. These methods compare the rate of LD decay around a single nucleotide polymorphism in contrast to the background decay rates or those of other ethnic groups. Several loci of possible positive selection have been detected by these methods based on the SNP datasets in HapMap Phase II [SRH$^+$02] and 1000 Genomes Project [GAS$^+$13] for European, African and East Asian populations.

Since long range haplotypes are observed in big LD blocks more frequently, the LD block formation and LRH might provide similar information for positive selection. In the meantime, denser datasets such as sequenced polymorphism data tend to show big LD blocks containing discontinuous regions of uncorrelated SNPs within the block region, which might affect LD decay rates estimated for LRH, iHS or XP-EHH [SVF$^+$07]. In this study, we examine the LD blocks and LD decay patterns of loci reported for possible positive selection in Sabeti et al. [SVF$^+$07] which were based on HapMap Phase II data of CEU, AFR and CHB+JPT populations. We use 1000 Genomes Project data [C$^+$12] of three major population subdivisions - European (EUR), East Asian (EAS), African (AFR) - to compare LD blocks and LD decays. We contrast the LD block boundaries of different populations obtained by a previously developed LD block partition algorithm Big-LD. We also obtain test results of XP-EHH [SVF$^+$07] and CMS [GAS$^+$13] to compare the LD block based measures and LD decay based measures. To calculate the XP-EHH rank scores of the reported sites in Sabeti et al. [SVF$^+$07] for 1000 Genomes Project data, we use The 1000 Genomes Selection Browser 1.0 [PDL$^+$13]. The CMS scores of the selected sites are obtained from the results reported in Grossman et al. (2013) where they construct the scores combining five tests including three LRH type measures (XP-EHH, iHS, ΔiHH) [SVF$^+$07, VKWP06] and two allele frequency based measures ($\mathrm{F_{st}}$, ΔDAF) [WC84]. We report that for several candidate regions, LD block

based comparisons provide independent information on LD structure with possibility of usage to find positive selection loci.

4.2 Methods

4.2.1 Previous methods: XP-EHH and CMS tests

XP-EHH (Cross-population Extended Haplotype Homozygosity) test detects selective sweeps in which the frequency of selected allele has rapidly increased reaching near or complete fixation due to positive selection. As a result, long-range haplotypes are formed near the selected allele [SVF$^+$07]. With respect to comparison of two populations, XP-EHH test calculates the decay of haplotype homozygosity (iHH, Integrated Haplotype Score) around the core SNP based on the probability that randomly chosen two chromosomes are identical (EHH, Extended Haplotype Homozygosity) for each population. The log of the ratio of iHH of two populations shows the population differentiation in terms of positive selection [SVF$^+$07].

To calculate the XP-EHH scores of candidate SNPs using the 1000 Genomes Project data, we used the 1000 Genomes Selection Browser 1.0 [PDL$^+$13] that provides diverse summary statistics for signatures of positive selection from the 1000 Genomes Project phase 1 data of CEU, CHB and YRI populations with 97, 85 and 88 individuals, respectively. We obtained the XP-EHH rank scores (negative log of p-value) and regarded the SNPs of maximum rank scores above 2.0 in one population compared to the other two populations as the evidence for positive selection.

Grossman et al. [GAS$^+$13] suggested a combined test of multiple tests, CMS, to detect positive selection. They combined independent five tests optimized to detect different patterns of signatures of selection, iHS, XP-EHH and ΔiHH for long haplotypes, and $\mathrm{F_{st}}$ and ΔDAF for high-frequency derived allele. The CMS test

combines the five tests in composite likelihood statistic, and this method is used to localize signals of natural selection within identified candidate regions. To identify new candidate regions, Grossman et al. [GAS+13] developed a new genome-wide CMS test called CMS_{GW}, and found that it has greater power than previous allele frequency spectrum based statistics and complements long range haplotype based methods. They applied CMS method to the region that is identified by CMS_{GW} and previous candidate region identified by long-haplotype methods [FBC+07], and suggested 412 fine-mapped signals of selection.

4.2.2 Comparison measure of LD block partitions

We calculated a similarity measure to compare LD blocks containing an index SNP obtained for two different populations. We used a modified version of a similarity measure for two partition results suggested by Fowlkes and Mallows [FM83]. For a given index SNP and two populations A and B to be compared, let X_A and X_B be the LD block regions (intervals on the chromosome) which include the position of the index SNP in population A and B, respectively. Let S_A and S_B be the sets of SNPs in the data of population A and B, located in the region X_A and X_B, respectively. We denote by x^A and x^B the total number of SNP pairs in the sets S_A and S_B, respectively. In addition, let x^{AB} be the number of pairs of SNPs in S_A which belong to the common region of two LD blocks X_A and X_B, and x^{BA} the number of pairs of SNPs in S_B which belong to the common region of two LD blocks X_A and X_B. Note that x^{AB} and x^{BA} might not be same if the lists of SNPs of each population are not same. Now the similarity measure $s(X_A, X_B)$ can be defined as follows:

$$s\left(X_A, X_B\right) = \sqrt{\frac{x^{AB}}{x^A} \times \frac{x^{BA}}{x^B}}.$$

4.2.3 Data

We obtained LD block construction results using Big-LD from 1000 Genomes Project phase 1 release 3 dataset of East Asian data (EAS, 286 individuals from CHB, CHS and, JPT populations), European data, (EUR, 379 individuals from TSI, IBS, FIN, CEU, and GBR populations), and African data (AFR, 246 individuals from YRI, LWK, and ASW populations) [C$^+$12] for several regions previously reported as for positive selection in Sabeti et al. [SVF$^+$07]. For the edge selection threshold for CLQ-D, we use $\theta = 0.5$ Sabeti et al.[SVF$^+$07] suggested 22 regions based on the long range haplotype tests along with some extra conditions: each suggested target SNP of selection should be a high-frequency derived allele, differentiated between populations, and common only in the selected population, and also identified as functional. Among 22 regions suggested as the strongest candidate regions for positive selection in Sabeti et al. [SVF$^+$07], we selected 11 regions that are near 16 genes: chr1:169Mb (BLZF1, SLC19A2), chr2:109 Mb (EDAR), chr2:136 Mb (RAB3GAP1, R3HDM1, LCT), chr2:178Mb (PDE11A), chr4:42Mb (SLC30A9), chr10:56Mb (PCDH15), chr15:49Mb (SLC24A5), chr15:64Mb (HERC1), chr16:76Mb (CHST5, ADAT1, KARS), chr17:59Mb (BCAS3), chr22:34Mb (LARGE). There were 30 individual SNPs which have shown strong evidence for positive selection within these 16 gene regions. We selected those SNPs as the index SNPs for our investigation. We used the phased genotype data in those regions after pruning SNPs with minor allele frequency less than 5% and removing deletions and insertions (indels).

To calculate the similarity scores of LD blocks in candidate regions, we chose a region of the LD block that contains each index SNP. If the similarity score calculated for two LD blocks obtained for two populations containing the index SNP is

below 0.8 and the ratio of the length of these LD blocks is below 0.8, we concluded that the LD block formation is different between two populations around the index SNP.

4.3 Results

In Table 4.1, we present the range and length of LD blocks including the index SNPs at the candidate positive selection sites for three populations (EAS, EUR, AFR) with marks for regions with the LD block similarity score$<$0.8 and LD block length ratio$<$0.8. In Table 4.2, we compare the populations detected as positive selection by different sources for candidate regions: 1) Sabeti et al. [SVF$^+$07] using XP-EHH combined with extra conditions applied to HapMap Phase II data, 2) Big-LD applied to 1000 Genomes Project data, 3) maximum XP-EHH scores calculated from 1000 Genomes Positive Selection Brower 1.0, and 4) Grossman et al. [GAS$^+$13] using CMS statistic applied to 1000 Genomes Project data.

The decision based on LD block comparison using 1000 Genomes Project dataset agrees with Sabeti et al. (2007) using HapMap dataset at several loci such as chr2:109 (EDAR), chr2:136 (RAB3GAP1. R3HDM1, LCT), chr10:56 (PCDH15), chr22:34 (LARGE).

In the region of EDAR gene (chr2:109329354$\sim$109540036), the LD block including the index SNP rs3827760 found in East Asian populations is 2.3 times longer than the blocks found in the other populations and the LD block similarity scores are 0.56 and 0.54 compared to European and Africans, respectively. Also, the maximum XP-EHH scores between CHB and CEU, and CHB and YRI obtained by the 1000 Genomes Positive Selection Browser are 2.62 and 3.85 respectively. However, this region was not found positively selected by Grossman et al.

Table 4.1: The LD block regions including the target index SNPs found in the candidate regions for positive selection.

SNPid	Gene	Pop	Size	Length (N. of SNPs)	Start (bp)	End (Mb)	(Mb)
chr1:169	rs1028180, rs3862937	BLZF1, SLC19A2	EAS*	635	282075	169.171	169.454
			EUR†	938	365025	169.089	169.454
			AFR	352	141309	169.309	169.45
chr2:109	rs3827760	EDAR	EAS*†	208	210683	109.329	109.54
			EUR	204	92371	109.448	109.54
			AFR	317	91091	109.449	109.54
chr2:136	rs17261772	RAB3GAP1	EAS	182	181594	135.758	135.94
			EUR*†	623	531335	135.758	136.29
			AFR†	1282	630221	135.757	136.387
chr2:136	rs1446585	R3HDM1	EAS†	705	605274	135.942	136.547
			EUR*	517	496045	136.291	136.787
			AFR	186	92777	136.387	136.48
chr2:136	rs4988235	LCT	EAS	382	219400	136.555	136.775
			EUR*†	517	496045	136.291	136.787
			AFR	483	153973	136.495	136.649
chr2:178	rs3770005	PDE11A	EAS*	266	79774	178.471	178.551
			EUR*†	670	318757	178.221	178.54
			AFR	356	65771	178.473	178.539
chr4:42	rs1047626, rs2660326, rs3827590, rs3827591, rs4861155	SLC30A9	EAS*	30	104697	41.983	42.088
			EUR†	442	211512	41.976	42.188
			AFR	389	121939	41.985	42.107
chr10:56	rs16905686, rs4935502	PCDH15	EAS*†	161	90039	55.87	55.96
			EUR	176	29452	55.947	55.976
			AFR	124	25948	55.944	55.97
chr15:49	rs1426654	SLC24A5	EAS	110	127548	48.39	48.517
			EUR*	NA	NA	NA	NA
			AFR	166	128270	48.39	48.518
chr15:64	rs10851731, rs2229749, rs2272209, rs2228511, rs6494428, rs16947373	HERC1	EAS*†	303	355711	63.781	64.136
			EUR	425	244328	63.893	64.137
			AFR†	850	346577	63.791	64.137
chr16:76	rs2242406, rs3743599, rs6834	CHST5, ADAT1, KARS	EAS*	265	177956	75.522	75.7
			EUR†	540	314622	75.523	75.837
			AFR*	661	175976	75.524	75.7
chr17:59	rs9303429	BCAS3	EAS†	396	350394	58.882	59.232
			EUR*	395	221876	58.813	59.035
			AFR	200	71975	58.902	58.974
chr17:59	rs6504005, rs6504010	BCAS3	EAS†	396	350394	58.882	59.232
			EUR*	395	221876	58.813	59.035
			AFR	116	54127	58.977	59.032
chr22:34	rs1573662	LARGE	EAS	27	19725	34.27	34.29
			EUR	38	16503	34.28	34.297
			AFR*†	92	24976	34.28	34.305
chr22:34	rs5999077	LARGE	EAS	44	23079	34.129	34.152
			EUR†	77	29462	34.122	34.151
			AFR*†	108	25825	34.123	34.149
chr22:34	rs1013337	LARGE	EAS	223	70921	34.153	34.224
			EUR	300	74788	34.152	34.227
			AFR*	249	73454	34.152	34.226

* Positive selection suggested by Sabeti et al. [SVF+07].

† LD block similarity score$<$0.8 and LD block length ratio$<$0.8.

Table 4.2: Comparisons of populations suggested for positive selection by different sources

Region	rsID (bp)	Gene	Populations on selection			
			XP-EHH [SVF^{+}07]	Big-LD	1000G Positive Selection Browser 1.0	CMS [GAS^{+}13]
chr1:169088679∼169453703	rs1028180 (169345868) rs3862937 (169436678)	BLZF1 SLC19A2	CHB+JPT	EUR	-	-
chr2:109329354∼109540036	rs3827760 (109513601)	EDAR	CHB+JPT	EAS	CHB	-
chr2:135756506∼136786651	rs17261772 (135911422) rs1446585 (136407479) rs4988235 (136608646)	RAB3GAP1 R3HDM1 LCT	CEU	EUR	CEU	CEU
chr2:178220864∼178550624	rs3770005 (178528874)	PDE11A	CEU, CHB+JPT	EUR	CEU, CHB	-
chr4:41976129∼42187640	rs1047626 (42003671) rs2660326 (42007071) rs3827590 (42031437) rs3827591 (42031472) rs4861155 (42032024)	SLC30A9	CHB+JPT	EUR	CHB	YRI
chr10:55869622∼55976483	rs16905686 (55949151) rs4935502 (55955444)	PCDH15	CHB+JPT	EAS	CHB	CHB+JPT
chr15:48389924∼48518193	rs1426654 (48426484)	SLC24A5	EUR	NA	CEU	CEU
chr15:63780762∼64137238	rs10851731 (63922752) rs2229749 (63937209) rs2272209 (63953153) rs2228511 (63954029) rs6494428 (63991725) rs16947373 (63998315)	HERC1	CHB+JPT	EAS, AFR	CHB	CHB+JPT
chr16:75522322∼75837352	rs2242406 (75574030) rs3743599 (75646576) rs6834 (75661803)	CHST5 ADAT1 KARS	CHB+JPT, YRI	EUR	-	CHB+JPT
chr17:58813056∼59232365	rs9303429 (58965400) rs6504005 (58986037) rs6504010 (59018306)	BCAS3	CEU	EAS	CEU, CHB	CEU
chr22:34121523∼34226847	rs1573662 (34289111) rs5999077 (34134109) rs1013337 (34211302)	LARGE	YRI	AFR	YRI	YRI

[GAS+13]. The EDAR gene encodes the key receptors for the ligands in ectodysplasin (EDA) pathway [BHP+08, SVF+07]. Several studies found the evidence of positive selection in this region based on allele frequency spectrum [WHC+07]. Other studies found the evidence of positive selection based on the population differentiation [AER+04] along with the haplotype structure [SVF+07] in this region. EDAR is known to have a role in development of hair follicles, teeth and exocrine glands [BHP+08, SVF+07]. The index SNP rs3827760 in EDAR region is a non-synonymous SNP which substitutes a valine to alanine at position 370 of the amino sequence (V370A). In 1000 Genomes Project dataset, the frequency of V370A is near fixation (87%) in East Asian whereas the frequency is l.1% and 0.3% in European and African populations.

RAB3GAP1, R3HDM1, and LCT genes all are located in the chr2:136 region. In this region, the LD block containing the index SNP rs4988235 near LCT gene in European populations is 2.3 and 3.2 times longer than the blocks in East Asian and African populations, and the LD block similarity scores are 0.46 and 0.49 compared to East Asian and African populations, respectively. In the region covered by LD blocks constructed by Big-LD (chr2:135756506∼136786651), the maximum XP-EHH scores between CEU and CHB, and between CEU and YRI obtained by 1000 Genomes Selection Browser 1.0 are 4.50 and 5.07 respectively. Also Grossman et al. [GAS+13] found evidence for positive selection in this region using CMS statistic. Mutations in LCT gene are associated with congenital lactase deficiency [SVF+07].

In the PCDH15 region, the LD block length for East Asian populations is three times the blocks in the other two populations and the LD block similarity scores of East Asians are 0.19 and 0.29 compared to the other two populations. In the region of LD blocks containing index SNPs (chr10:55869622∼55976483), the maximum

XP-EHH scores between CHB and CEU, and between CHB and YRI are 2.92 and 2.90 respectively, and this region overlaps with the candidate region of positive selection in CHB+JPT population detected by CMS statistic [GAS$^+$13]. The role of PCDH15 is for retinal and cochlear function [AER$^+$04] and development of inner-year hair cells [GAS$^+$13]. The frequency of the derived allele of rs16905686 in East Asian is 84% whereas it is 13% in the other two populations.

The LD block including rs1573662 in the LARGE gene region is 1.3∼1.5 times longer in African populations than in the other two populations. The LD block similarity scores of African populations are 0.74 compared to Europeans and 0.14 compared to East Asians for the region including the index SNP rs1573662. In the region of LD blocks containing index SNPs (chr22:34121523∼34226847), the maximum XP-EHH scores between YRI and CEU, and between YRI and CHB obtained by 1000 Genomes Selection Browser 1.0 are 2.99 and 2.10 respectively. Also, this region overlaps with the candidate region of positive selection in YRI population detected by CMS statistic [GAS$^+$13]. The region of LARGE gene is well known to be linked to the Lassa fever virus [AST$^+$12]. In Lassa, Nigeria, the 20% of the population is exposed to this disease [AST$^+$12, SVF$^+$07]. Many researchers have hypothesized that the Lassa fever epidemic pressured the selection at LARGE [AST$^+$12, RB03, SVF$^+$07]. The frequency of the derived allele of rs1573662 is 24% in Africans, but is 0.4% in Europeans. The allele is not observed in East Asians.

The SLC24A5 gene region is known to have a role in pigmentation and skin colors [SVF$^+$07]. In European populations of 1000 Genomes Project data, the frequency of the derived allele of the index SNP rs1426654 approaches 99.7% whereas it is observed to be 1.2% and 7.4% of East Asians and Africans. Because of low minor allele frequency of the SNPs in the region around the index

SNP in European populations, we could not detect the LD block containing the index SNP for this population. In the region that contains the index SNP rs1426654 (chr15:48389924∼48518193), the maximum XP-EHH scores between CEU and CHB, and between CEU and YRI obtained by 1000 Genomes Selection Browser 1.0 are 4.19 and 3.36 respectively. This region, also, overlaps with the region of positive selected in CEU population found by Grossman et al. [GAS+13]. Some findings based on LD block comparisons were inconsistent with the results reported by Sabeti et al. [SVF+07]. Our results indicate that populations different from the previously suggested population as positively selected for regions of BLZF1-SLC19A2 and CHST5-ADAT1-KARS and HERC1. In HERC1 region, the LD blocks containing the index SNPs are found to have shorter lengths only in Europeans. On the other hand, the LD heatmaps show that the overall LD between the SNPs in the region is stronger in EAS than in the other populations, while the blocks are formed rather similarly for three populations (Figure 4.1A). The LD decay plot of the index SNP rs16947373 shows that the SNPs strongly linked to the index SNP are observed to be located at similar distance in three populations yet they are more frequent in EAS population (Figure 4.1B). The XP-EHH score plot of all SNPs in this region clearly shows that in the CHB population SNPs show strong signals for LD decay rates (Figure 4.1C). The maximum XP-EHH scores between CHB and CEU, and between CHB and YRI obtained for this region are 3.06 and 2.28 respectively. Also, this region overlaps with the candidate region of positive selection in CHB-JPT population detected by CMS statistic [GAS+13]. This case shows that the differences in the strength of LD may result in similar LD block formations between different populations by Big-LD but also different values for the LD decay measures. For BLZF1-SLC19A2 region, and CHST5-ADAT1-KARS region, the maximum XH-EHH scores obtained for 1000 Genomes Project did not indicate any

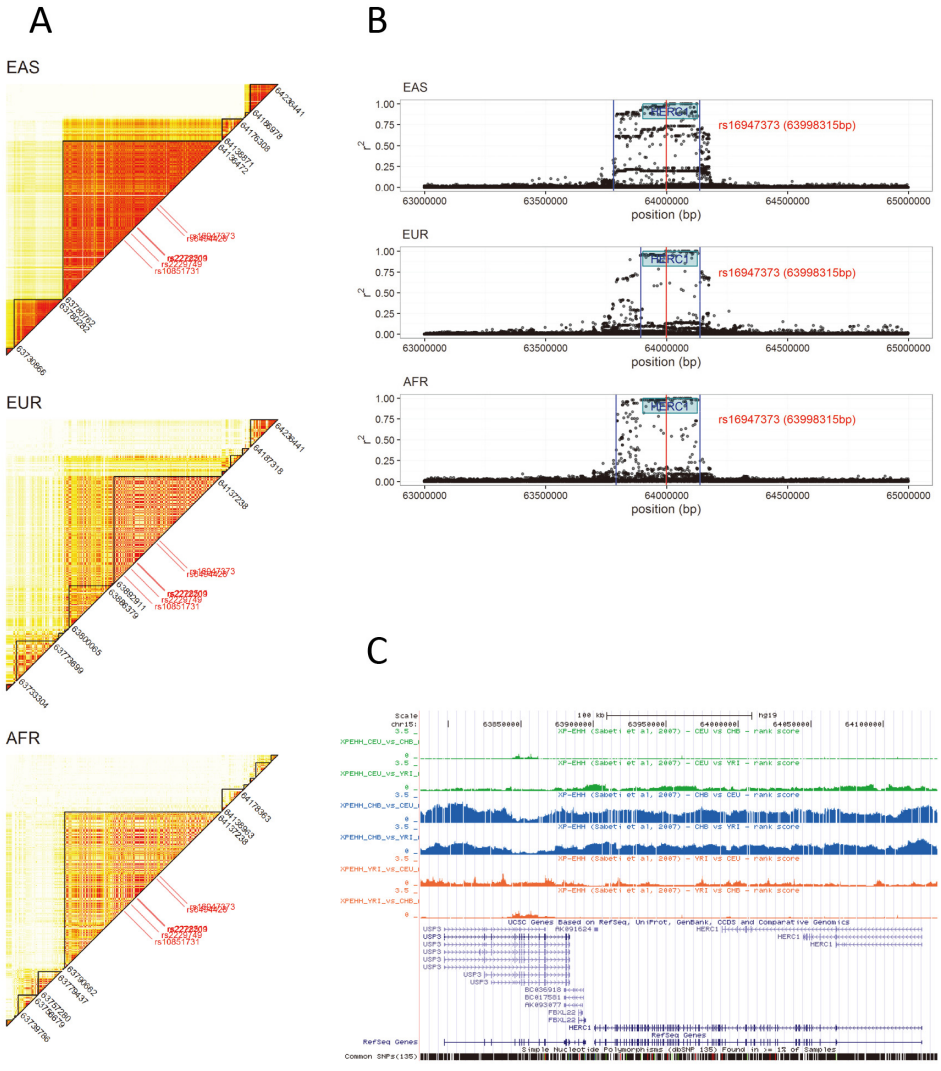


Figure 4.1: (A) LD heatmaps of HERC1 region for three populations. LD block boundaries are marked by solid lines and the index SNPs in this region is marked in red. (B) LD decay map around the index SNP rs16947373. The r^2 values of the SNPs away from the index SNP (red vertical line) is plotted from the centre. The LD block containing the index SNP is marked in blue vertical line. (C) XP-EHH rank score plot obtained from the 1000 Genomes Selection Browser 1.0

particular population in the aspect of positive selection. For CMS statistic, CHB-JPT was suggested as positively selected in CHST5-ADAT1-KARS region, which is also inconsistent with the LD block analysis or findings of Sabeti et al. [SVF+07].

In SLC30A9 region, the LD block found in European populations is the longest while the East Asian population has been suggested as positive selection in Sabeti et al. [SVF+07]. The LD block similarity scores of Europeans are 0.55 and 0.42 compared to East Asians and Africans, respectively. The LD heatmaps show that the block formed around the index SNPs in European populations is longer compared to the other populations (Figure 4.2A). However, the LD decay of index SNP rs4861155 (Figure 4.2B) and the XP-EHH score plot for this region (Figure 4.2C) show that the LD decay is slower in East Asian populations. In the results of the 1000 Genomes Selection Browser 1.0, the XP-EHH scores of four SNPs among the suggested five index SNPs do not show the significant signal of positive selection. However, these index SNPs belong to one of the candidate regions suggested by Grossman et al. [GAS+13] for the selection in YRI. The frequencies of the derived allele of the index SNP rs1047626 are 95% and 76% in East Asian and European populations, and 11% in African populations. In other studies, possible positive selection in East Asians for SLC30A9 has been also suggested [EEM+15, ZLT+15]. SLC30A9 encodes the Zinc transporter protein ZNT9 and has been suggested to be positively selected in relation to the zinc distribution in soils and crops in different geographical regions [ZLT+15]. The LD block based analysis, LRH based results and a combined statistic CMS suggest different population as positive selection, which might be due to complex population history calling for more elaborated analysis. In PDE11A region, LD block based comparisons indicate that European is possibly positively selected while East Asian and European both have been suggested as positively selected in this region by Sabeti et al. [SVF+07]. The LD

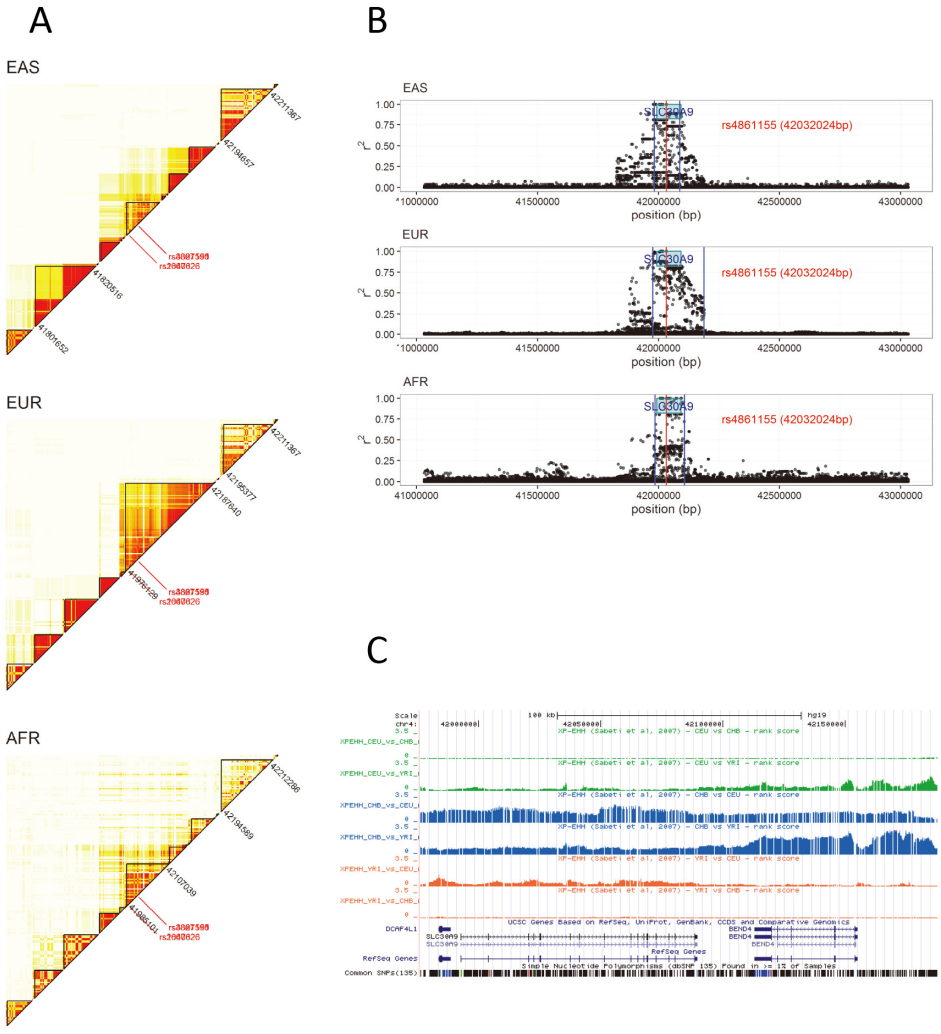


Figure 4.2: (A) LD heatmaps of SLC30A9 region for three populations. LD block boundaries are marked by solid lines and the index SNPs in this region is marked in red. (B) LD decay map around the index SNP rs4861155. The r^2 values of the SNPs away from the index SNP (red vertical line) is plotted from the centre. The LD block containing the index SNP is marked in blue vertical line. (C) XP-EHH rank score plot obtained from the 1000 Genomes Selection Browser 1.0

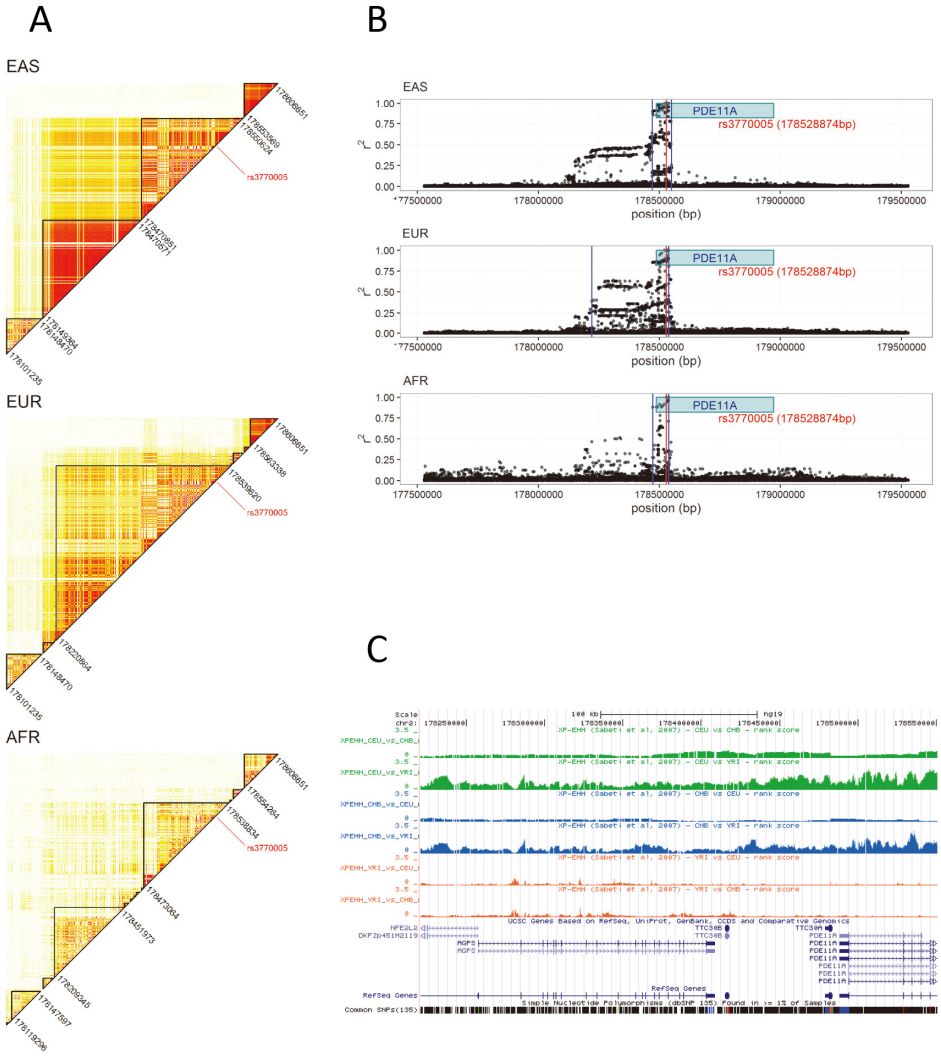


Figure 4.3: (A) LD heatmaps of PDE11A region for three populations. LD block boundaries are marked by solid lines and the index SNPs in this region is marked in red. (B) LD decay map around the index SNP rs3770005. The r^2 values of the SNPs away from the index SNP (red vertical line) is plotted from the centre. The LD block containing the index SNP is marked in blue vertical line. (C) XP-EHH rank score plot obtained from the 1000 Genomes Selection Browser 1.0

heatmaps show an LD block is formed in wider range in European populations (Figure 4.3A). Meanwhile, the LD decay plot shows stronger LD for surrounding SNPs with the index SNP rs3770005 in CHB and CEU (Figure 4.3B, 4.3C). The frequencies of derived alleles of the index SNP are 70% and 72% in East Asian and European populations, but it is 9.8% in African populations. In the LD block region (chr2:178220864∼178550624) which contains index SNP rs3770005 constructed by Big-LD, the maximum XP-EHH scores obtained by 1000 Genomes Selection Browser 1.0 between CEU and YRI, and between CHB and YRI are 2.94 and 3.20 respectively. This region does not overlap with any of the candidate regions suggested by Grossman et al. [GAS+13]. The mutations in PDE11A cause Cushing disease and adrenocortical hyperplasia [Str08].

In BCAS3 region, the LD block found in East Asian populations is the longest and the LD block similarity scores are 0.27 and 0.36 compared to European and African populations, but Sabeti et al. [SVF+07] suggested European population as for the positive selection. When we examined the LD heatmaps, we found the LD patterns in both European and Asian are similar (Figure 4.4A). In this region, we could observe that the LD decay of three index SNPs is slower in East Asian and European populations, with slowest decay in East Asians (Figure 4.4B). The frequencies of rs9303429 allele are 87% and 83% in East Asian and European populations while it is 19% in Africans. In the LD block region containing index SNPs, the maximum XP-EHH scores from 1000 Genomes Selection Browser 1.0 between CEU and YRI, and between CHB and YRI are 4.75 and 6.00 respectively. Figure 4.4C shows the results of 1000 Genomes Selection Browser 1.0 in this region, and similar patterns between the CEU and CHB populations in the XP-EHH scores plots can be observed. On the other hand, this region overlaps the region of positive selection in CEU population suggested by Grossman et al. [GAS+13]. The BCAS3

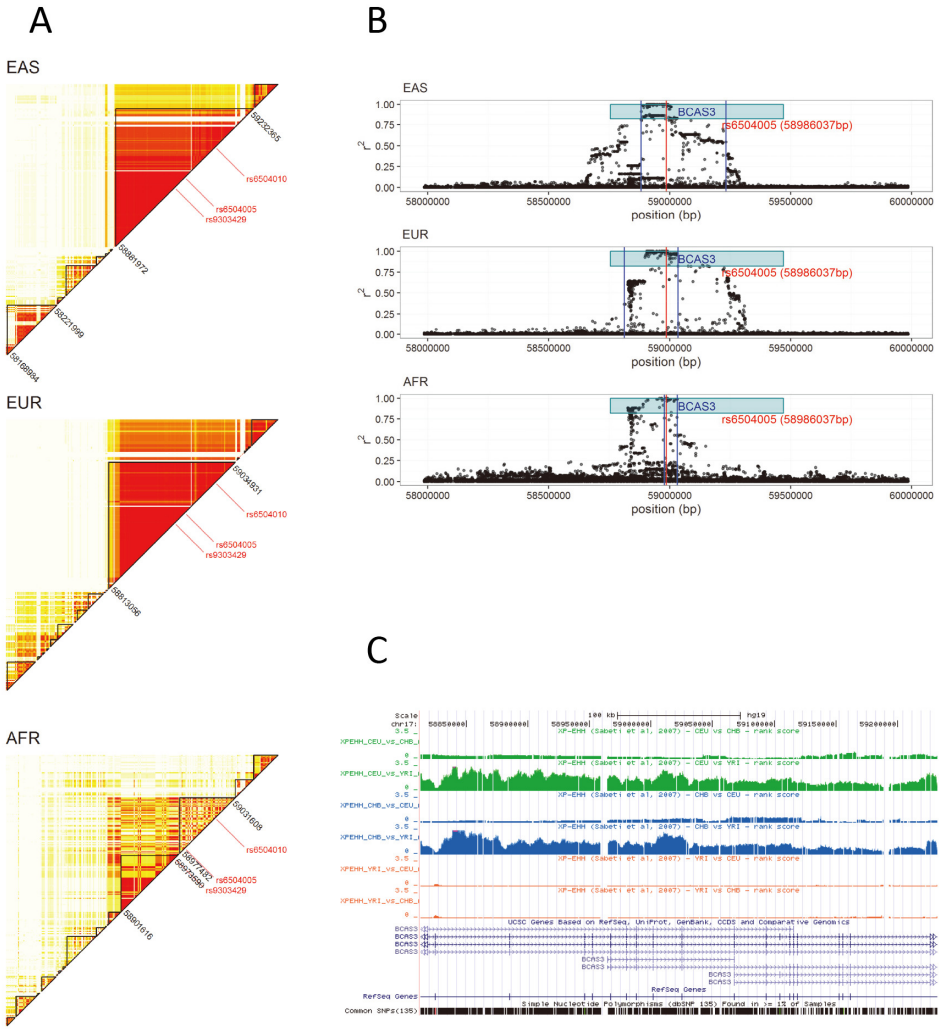


Figure 4.4: (A) LD heatmaps of BCAS3 region for three populations. LD block boundaries are marked by solid lines and the index SNPs in this region is marked in red. (B) LD decay map around the index SNP rs65504005. The r^2 values of the SNPs away from the index SNP (red vertical line) is plotted from the centre. The LD block containing the index SNP is marked in blue vertical line. (C) XP-EHH rank score plot obtained from the 1000 Genomes Selection Browser 1.0

gene is a protein coding gene and related to the estrogen response. This gene is amplified and over-expressed in breast cancer cells [BMW+02], and has been reported for association with breast cancer and coronary artery diseases [BHG+16].

4.4 Conclusions

We examined the LD block structures attained by the LD block construction algorithm Big-LD in the regions including selected candidate loci of positive selection suggested in Sabeti et al. [SVF+07] using the 1000 Genomes Project data of three populations (East Asian, European, and African). We could find LD block patterns in genes EDAR, LCT, PCDH15, and LARGE consistent with the results in Sabeti et al. [SVF+07] and confirm the positive selection in these loci. We observed that in HERC1, SLC30A9, PDE11A and BCAS3 regions, the LD block based analysis provides different information worth considering in detection of positive selection. Based on all evidences from this study, we suggest that SLC30A9, PDE11A and BCAS3 regions have been positively selected in both European and East Asians with possible discrepancy between two populations.

Big-LD algorithm provides information about LD structure of multiple non-consecutive SNPs in strong LD by constructing LD blocks that contain "holes" [WP03]. This information can add extra contribution to the findings drawn from LD decay analysis or allele frequency analysis. To investigate positive selection with greater power, we might want to combine LD block comparison method with the other statistical tests which designate a single-SNP as a core of selection procedure.

Chapter 5

Sparse hypergraph partitioning

5.1 Motivation and background

The Big-LD algorithm is designed to partition SNP sequence data into blocks. The SNP sequence data have a particular characteristic such that SNPs are ordered by their positions in the chromosome. In addition, due to population genetic theory, there is little chance that distant SNPs are in strong LD. If we represent SNPs by a graph based on their pairwise LD, distant SNPs are usually not joined by an edge and this restricts the number of edges.

In this chapter, we try to generalize the clustering problem of SNP sequence data to the situation of a general dataset without any positional ordering, but with the community membership or common attributes information available. In the Big-LD algorithm, consecutive SNPs in the range of clusters are designated by an interval, and an interval graph is constructed based on the overlap between those intervals. For a general clustering problem not considering the ordering of the objects, the concept of intervals cannot be applied. Instead of intervals, the community membership and their overlap structures can be modeled by a hypergraph using hyperedges.

In the case of the Big-LD algorithm for SNP sequence data, the number of intervals is smaller than the total number of SNPs. Hence, the interval graph has fewer vertices, resulting in the rapid computation of Big-LD execution. This shows the benefit of the multi-level modeling approach used for Big-LD. For a generalized

clustering problem with community membership information, we will assume that the number of hyperedges is less than or equal to the number of vertices. We call such a hypergraph a "sparse hypergraph" in this study.

When we focus on common attributes or memberships of objects rather than the pairwise relationships as we did for Big-LD regarding LD bins, the hypergraph modeling can represent the data structure naturally. For example, co-authorship networks can be modeled as a simple graph, where vertices represent authors and two vertices are joined by an edge if they have a co-author relationship [New04]. On the other hand, using a hypergraph, we can model the information of co-authors by representing the co-authors of a work by a hyperedge [HZPJ]. For genetic data, we can group genes, proteins, or phenotypes sharing common biological properties and represent them by hypergraphs [GDG14, THK09, KHZ14, RPT10]. Social media information, such as social tagging systems, can be transformed into a hypergraph by grouping objects having common attributes and used for social network analysis or recommendation system design [ZL10, TBCH11, BTC$^+$10].

Attempts have been made to partition data using hypergraph modeling. The multi-level approach is the one of the most widely used hypergraph partitioning strategies, and this approach consists of three phases: coarsening, initial partitioning, and uncoarsening. This method has been applied to various problems such as VLSI (very-large-scale integration) design circuits [KAKS99, AHK98, LJ15, KK98b, AHK98, KK00, KAKS99, LJ15, CDKU12]. Another approach is spectral clustering. Shi and Malik [SM00] suggested a spectral clustering algorithm based on the normalized cut approach which operates on an undirected graph to solve the problems of image segmentation. Generalized spectral clustering techniques for hypergraphs have been suggested by Zhou et al. [ZHS06]. A hypergraph can be transformed into an undirected graph by joining every two vertices in the same hyperedge

by an edge, then a spectral clustering algorithm can be applied to the vertex set of the hypergraph [OB12, ALZM$^+$05]. Bulò and Pelillo [BP09] applied a game theory by considering the hypergraph clustering problem as a multi-player non-cooperative clustering game. Liu et al. [LLY15] suggested an algorithm, called dense subgraph partitioning (DSP), which partitions a hypergraph by detecting dense subgraphs in a greedy manner and the number of partitions are automatically decided. However, most hypergraph clustering methods require parameters such as the number of partitions (multi-level approach [AHK98, KK98b, KK00, Kar03, KAKS99], spectral approach [ZHS06, OB12, ALZM$^+$05]), or impose the constraints such as the balance of the weights of clusters (multi-level approach [KK98b, KAKS99, AHK98, LJ15]) or the size of hyperedges (game theory approach [BP09], spectral approach [OB12, ALZM$^+$05]). There exist cases that need a pre-decided number of clusters, size of hyperedges, or balanced size of hyperedges of input data. However, when we attempt to cluster real-world data, methods with such limitations might not always produce desirable results. In the problem of partitioning SNP sequences into LD blocks, limiting the size of LD bins or LD blocks or imposing a balance between the sizes of LD bins or LD blocks might twist the big LD block structures we intend to find.

In this chapter, we develop several hypergraph partitioning algorithms without constraints on the size of hyperedges, number of final partitions, and balance between sizes of partitions assuming the hypergraph is sparse in that the number of edges is less than the number of vertices.

5.2 Related works

A large number of hypergraph partitioning methods have been developed. In this section, we review the multi-level hypergraph partitioning method and the spectral hypergraph partitioning method that are regarded as the most widely used and important k-way partitioning methods. We also review the DSP method that decides the number of partitions automatically in contrast to the k-way partitioning method. In addition, k-clique clustering is reviewed, which is used to find overlapping clusters in networks, and in this work, we suggest a notion of "max-clique adjacency" based on the "k-clique adjacency" related to k-cliques.

5.2.1 Multi-level hypergraph partitioning

Multi-level hypergraph partitioning is one of the most widely used hypergraph clustering methods in fields such as VLSI design and data mining [KK98b, AHK98, KK00, KAKS99, LJ15, CDKU12]. There are two approaches in multi-level hypergraph partitioning: recursive bisection [KAKS99, AHK98, LJ15], and direct k-way partitioning [KK98b, KK00]. The goal of both methods is to partition the vertices of a hypergraph into k parts, such that objective functions are optimized under the requirement about the weight of each partition. The recursive bisection approach generates a bipartition of the original hypergraph, and then recursively applies the bisecting procedure to each part independently. The direct k-way partitioning calculates k partitions of a hypergraph directly. Basically, this method aims to partition the vertex set into k parts of equal weights as close as possible.

The multi-level hypergraph partitioning method consists of three phases: coarsening, initial partitioning, and uncoarsening. During the coarsening phase, hypergraphs are successively contracted by merging two or more vertices, and a sequence

of successively smaller hypergraphs are constructed. The purpose of the coarsening stage is to reduce the sizes of hyperedges, and this helps to refine hyperedges with a large number of vertices when they belong to different partitions. There are a number of schemes for merging vertices to obtain the next level hypergraph, and we introduce most representative approaches : edge-coarsening, hyperedge-coarsening, and first-choice edge coarsening.

Edge coarsening is the simplest way to merge the vertices connected by hyperedges. In this scheme, for each vertex v the algorithm visits, unmatched vertices that belong to hyperedges incident to v are selected and pairs of vertices with the largest weight are chosen as a group [KAKS99]. In the *hyperedge-coarsening* scheme, an independent set of hyperedges is selected and the vertices in each hyperedge are contracted together. First, all hyperedges are sorted by non-increasing order of hyperedge weights. The algorithm visits hyperedges in that order, and selects a hyperedge that does not share any vertex with already selected hyperedges. All the vertices in the selected hyperedge are contracted together. Because more than two vertices are grouped, this approach speeds up the coarsening phases [KAKS99]. In addition to the hyperedge-coarsening procedure, for each hyperedge we can merge vertices that do not belong to the already contracted hyperedges. This scheme is named *modified hyperedge-coarsening* [KAKS99]. Edge and hyperedge coarsening both find independent groups for contracting. These approaches may destroy the natural cluster structure in the hypergraph, so the *first-choice edge-coarsening* scheme was developed to overcome this problem. The first-choice edge-coarsening scheme adopts a relaxed requirement compared with the edge-coarsening scheme in that a vertex can be matched with an already matched vertex by breaking the existing tie if the new pair of vertices is connected with the largest weight [Kar03].

During the initial partitioning phase, bisections or k-way partitions of the smal-

ler hypergraphs are computed. Since the coarsest hypergraph constructed by the coarsening phase has few vertices, the execution time for partitioning tends to be small. Since the coarsest hypergraph may not reflect the structure of the original hypergraph, the algorithm finds partitions that have small cut and also satisfy a balancing constraint. One method of initial partitioning is to find a random bisection that satisfies the balance constraint. Another method is to start from a randomly selected vertex seed and then grow the part by assigning vertices around the vertex seed to the part until the part contains half of the vertices [Kar03].

During the uncoarsening phase, the partitioning of coarser hypergraphs is projected to the next-level finer hypergraph, and a reduction in the number of hyperedges cut by partitions is implemented using a refinement algorithm. The Fiduccia–Mattheyses (FM) algorithm [FM88], which is the basis of the various refinement algorithms, is an iterative method. In each iteration, it detects a subset of vertices in each partition to move them to other partitions to improve the quality of partitioning [KAKS99].

The `hMETIS` is one of the most widely used programs to partition a hypergraph into k parts using multi-level hypergraph partitioning methods. It contains both recursive bisection methods (shmetis and hmetis) and direct k-way partitioning (khmetis) [KAKS99, KK98b]. The program provides options for various coarsening, uncoarsening and refinement schemes based on the FM algorithm.

5.2.2 Spectral clustering

Another k-way hypergraph clustering method is the spectral clustering approach. To solve the image segmentation problem, Shi and Malik [SM00] suggested a spectral clustering method based on simple graphs that minimizes normalized cut by solving the generalized eigenvalue system. A simple graph $G = (V(G), E(G))$ can be

partitioned into two disjoint sets A and B, and the nomalized cut is defined as

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V(G))} + \frac{cut(A,B)}{assoc(B,V(G))}$$

such that $cut(A,B) = \sum_{u\in A, v\in B} w'_G(u,v)$ and $assoc(A,V(G)) = \sum_{u\in A, v\in V(G)} w'_G(u,v)$ where $w'_G(u,v)$ is the edge weight between u and v. The exact minimizing normalized cut is NP-complete, however we can find an approximate solution by solving the generalized eigenvalue system. Let $d(v_i) = \sum_{v_j\in V(G)} w'_G(v_i,v_j)$ be the total connection from a vertex v_i to all other vertices, $\mathbf{D}$ be a $|V(G)| \times |V(G)|$ diagonal matrix with $\mathbf{D}[i,i] = d(v_i)$, and $\mathbf{W}$ be a $|V(G)| \times |V(G)|$ matrix with $\mathbf{W}[i,j] = w'_G(v_i,v_j)$. Then, the eigenvector associated with the second smallest eigenvalue of the eigenvalue system

$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}\mathbf{z} = \lambda\mathbf{z},$$

where $\mathbf{z} = \mathbf{D}^{\frac{1}{2}}\mathbf{y}$, is the real valued solution to our normalized cut problem.

For hypergraph partitioning, a generalized version of the normalized cut for a simple graph can be defined, and we can formulate the optimization problem for hypergraph partitioning in a similar manner to the spectral partitioning of a graph. To obtain k-way partitions, the K-means algorithm [HW79] can simply be run on a matrix whose columns are k eigenvectors associated with the k smallest eigenvalues [ZHS06]. On the other hand, we can partition a hypergraph by transforming the hypergraph H into a graph G where $V(G) = V(H)$ and $v_i, v_j \in V(G)$ are joined by an edge if and only if $\{v_i, v_j\} \subset e$ for $e \in E(H)$. Then we can apply the spectral clustering method for graphs and obtain the partitioning results [OB12, ALZM$^+$05].

5.2.3 Dense subgraph partitioning

DSP [LLY15] is a method to partition a hypergraph (or graph) with positive edge weights into dense subgraphs. DSP is non-parametric partitioning and automatically decides the number of partitions. The density of a hypergraph $H = (V(H), E(H))$ is defined to be $\rho(H) = \frac{w'(H)}{|V(H)|}$ where $w'(H) = \sum_{e \in E(H)} w'_H(e)$. Then there is a maximal densest subgraph of H that is called the *core subgraph* denoted as $CS(H)$. For two subsets $U, S \in V(H)$, the conditional total weight of an induced subgraph $H[U]$ conditioned on an induced subgraph $H[S]$ is defined as $w'(H[U]|H[S]) = w'(H_{U \cup S}) - w'(H[S])$, and the conditional density of $H[U]$ conditioned on $H[S]$ is defined as $\rho(H[U]|H[S]) = \frac{w'(H[U]|H[S])}{|U|}$. A conditional core subgraph conditioned on a subgraph $H[S]$, denoted as $CCS(H|H[S])$, is the maximal subgraph whose conditional density reaches a maximum.

DSP is a method to sequentially partition a given hypergraph into a sequence of disjoint conditional core subgraphs. That is, we first find a core subgraph $H[V_1]$ of H and find a disjoint partition of the core subgraph whose density is equal to the density of the core subgraph, then we detect a conditional core subgraph $H[V_2]$ conditioned on $H[V_1]$ and find the disjoint partition of $H[V_2]$ whose conditional density is equal to the conditional density of $H[V_2]$. After we find a partition P_1, we repeat the same procedure on the graph $H(V \setminus P_1)$ and similarly find another partition P_2. In this way, the disjoint partitions of H can be found by repeating the procedure.

5.2.4 k-clique clustering

Numerous studies have investigated the clustering methods related to the notion of cliques. We review the k-clique clustering that has been the motive for the concept

of *max-clique adjacency* in this study. The concept of k-clique is utilized to reveal the overlapping structures of communities in real-world networks [PDFV05]. A *k-clique* is a complete subgraph of k vertices. There are few notions related to k-cliques [DPV05].

- Two k-cliques are *k-clique adjacent* if two k-cliques share $k-1$ vertices.
- A *k-clique chain* is the union of a sequence of adjacent k-cliques.
- Two k-cliques are *k-clique-connected* if they belong to a k-clique chain.
- A *k-clique percolation cluster* is a maximal k-clique-connected subgraph

The k-clique community (k-clique percolation cluster) finding algorithm suggested in [PDFV05] is as follows.

We first detect all maximal cliques $C_1, ..., C_n$ of a graph (or network) G. Then an $n \times n$ clique-clique overlap matrix M is constructed such that $M[i, j]$ is the number of common vertices of maximal cliques C_i and C_j, and diagonal entries are the sizes of the maximal cliques. To find k-clique communities, we first need to delete every non-diagonal entry of M smaller than $k-1$ and every diagonal entry of M smaller than k. A k-clique community is the union of maximal cliques that corresponds to the vertices of connected components of a graph whose adjacency matrix is the modified M. This method is used for an undirected graph with unweighted edges. For an edge-weighted graph, we can set a threshold w^* to keep edges with weights greater than the given threshold. Increasing k or w^* makes the communities smaller but more cohesive.

5.3 Hypergraph partitioning algorithms

5.3.1 Algorithm overview

We propose several new hypergraph partitioning algorithms for sparse hypergraphs using a multi-level approach. The algorithms consist of three phases.

(1) Construct an edge-weighted line graph of a hypergraph.

(2) Detect "dense sets" of the line graph based on the notion of maximal cliques.

(3) Construct an intersection graph with the dense-set information, and compose final clusters by finding the maximum weight independent set (MWIS) of the intersection graph.

For the Big-LD algorithm to partition SNP sequence data, we can consider any clique of the interval graph as a candidate for LD block because the union of incident intervals always forms a block. For a general partitioning problem where the ordering of the vertices does not matter and the resulting partitions need not be consecutive, we introduce a new method to suggest candidates for clusters.

For the second phase, we define the *edge-density* of a graph as the ratio of the number of edges in the graph to the number of vertices. A subgraph of the graph with relatively higher edge-density can be regarded as a candidate cluster subject to consideration. For hypergraph clustering, we find "dense sets" that are the vertex sets of a subgraph with relatively higher edge density. To decide the scope of dense set, we suggest the notion of *max-clique adjacency* and *near-maximal clique* analogous to k-clique adjacency introduced in section 5.2. We propose two different algorithms to detect the dense sets of the line graph, STRONG-CLQ and RELAXED-CLQ, for the second phase. STRONG-CLQ considers only maximal cliques and near-maximal cliques as dense sets, while RELAXED-CLQ allows more expanded dense sets including maximal cliques, near-maximal cliques, and merged maximal

cliques that are max-clique adjacent.

For the third phase, we propose two different methods to construct an intersection graph. We find dense sets of the line graph in the second phase. The union of hyperedges corresponding to the vertices of a dense set (denoted by the "candidate vertex set") of the line graph can be considered as a candidate cluster. The family of candidate vertex sets of a hypergraph can be taken as a vertex set of an intersection graph, and the family of dense sets can also be taken as a vertex set of an intersection graph. We propose two algorithms: CSET-CLST and QSET-CLST. The first clustering algorithm is based on the intersection graph of the family of the candidate vertex sets and the second algorithm is based on the intersection graph of the family of the dense sets of a line graph.

The definitions of the suggested notions and details of the algorithms are stated in later sections (5.3.2∼5.3.4).

5.3.2 Construction of the line graph of a hypergraph

Let H be the hypergraph in consideration for clustering (here, by clustering we mean partitioning). For a hypergraph H, we construct an edge-weighted line graph $L(H)$ where the set of vertices of $L(H)$ is the hyperedge set $E(H)$. For each edge $(e_i, e_j) \in E(L(H))$, we define its edge weight as

$$\sqrt{\frac{|e_i \cap e_j|}{|e_i|} \times \frac{|e_i \cap e_j|}{|e_j|}},$$

that is called an Ochiai coefficient [Och57, SS$^+$63] or Otsuka coefficient [Pet68]. If two hyperedges are represented as vectors, the Ochiai coefficient is identical to cosine similarity. The Ochiai coefficients emphasize similarity between two sets [CH69].

Algorithm 5.1: ADJLH : Implements a hypergraph into a matrix in terms of Ochiai coefficient

input : A list of hyperedges $\{e_1, ..., e_m\}$ of H
output: A matrix representing H

$A \leftarrow m \times m$ matrix;
for $i \in \{1, ..., m-1\}$ **do**
  for $j \geq i$ **do**
    if $j > i$ **then**
      $A[i,j] \leftarrow \sqrt{\frac{|e_i \cap e_j|}{|e_i|} \times \frac{|e_i \cap e_j|}{|e_j|}}$;
      $A[j,i] \leftarrow A[i,j]$;
    else
      $A[i,i] \leftarrow 0$;

Definition 8. *A graph is **bipartite** if its vertex set can be partitioned into two subsets A and B so that every edge has one end in A and one end in B; such a partition (A,B) is called a **bipartition** of the graph, and A and B its parts. If a bipartite graph G with bipartition (A,B) is simple and every vertex in A is joined to every vertex in Y, then G is called a **complete bipartite graph** and denoted by $K_{|A|,|B|}$.*

Observation 1. *For a hypergraph $H = (V(H), E(H))$, a complete bipartite graph $K_{k,l}$ cannot be a line graph of the hypergraph H if $lk > |V(H)|$.*

Proof. Suppose that there exist a hypergraph $H = (V(H), E(H))$ and its line graph is a complete bipartite graph $K_{k,l}$ where $lk > |V(H)|$. Let (A, B) be a bipartition of the complete bipartite graph $K_{l,k}$ where $A = \{v_1, ..., v_k\}$ and $B = \{w_1, ..., w_l\}$. We denote by e_i (respectively f_i) the hyperedge corresponding to v_i (respectively w_j). Since every vertex in B is adjacent to every vertex in A, $e_i \cap f_j \neq \emptyset$ for each $1 \leq i \leq k$, $1 \leq j \leq l$. Since there is no edge between any pair of vertices in A, $e_i \cap e_j = \emptyset$ if $i \neq j$. Thus, $|f_j| \geq |\bigcup_{i=1}^{k}(e_i \cap f_j)| = \sum_{i=1}^{k} |e_i \cap f_j| \geq k$. Since $f_i \cap f_j = \emptyset$ for $i \neq j$, $|\bigcup_{j=1}^{l} f_j| = \sum_{j=1}^{l} |f_j| \geq kl > |V(H)|$ and we reach a contradiction. □

5.3.3 Listing the dense sets of the line graph

For the edge-weighted line graph $L(H)$ of the hypergraph H, we now find "dense sets" based on the maximal cliques of the line graph to compose candidate clusters. These dense sets will become the vertices of an intersection graph constructed in the next procedure (5.3.4). Note that two hyperedges are joined by an edge even if they share relatively few elements in the previous line graph construction procedure. If we consider all edges, edges with relatively low weights might distort the underlying clustering structure and generate undesired candidate clusters. In addition, edge density affects the time complexity of the algorithm that finds the maximal cliques. Thus, to find reliable candidate clusters effectively, we introduce a threshold θ of the edge weight for valid edges. Before finding the "dense sets" of the line graph, we prune edges by deleting edges with weights below the threshold θ and let $L'(H)$ denote the line graph obtained by pruning the edges.

Now, we find all maximal cliques of the graph $L'(H)$ using the Bron–Kerbosh algorithm [BK73] suggested in [ELS10]. We define the *edge density* of a graph G as $|E(G)|/\binom{|V(G)|}{2}$. Clearly, the edge density of a maximal clique is 1. In maximal cliques, every pair of vertices is joined by an edge with weights above a given threshold θ, hence a maximal clique can be regarded as a strong cohesive community. However, it might be too strict to consider only maximal cliques as candidate clusters. In addition to the maximal cliques, if we could find more flexible yet reliable candidate clusters, it would improve the quality of the clustering results.

To find "dense sets" that are not maximal cliques but still sufficiently dense, we introduce the notions of *max-clique adjacency* and *near-maximal clique* by borrowing the notion of k-clique adjacency.

Definition 9. (Max-clique adjacency) *Two maximal cliques Q' and Q'' of a graph*

*G are **max-clique adjacent** if the two maximal cliques share $\min(|Q'|, |Q''|) - 1$ vertices.*

Definition 10. (Near-maximal clique) *A subset S of $V(G)$ is the **near-maximal clique** of G if the set S is a sub-clique of a maximal clique Q with size $|Q| - 1$, or a union of two maximal cliques Q' and Q'' that are max-clique adjacent.*

We can obtain sufficient dense sets of $L'(H)$ by finding near-maximal cliques.

Theorem 1. *For a near-maximal clique Q of a graph G, the edge density of $G[Q]$ is greater than or equal to $\frac{2}{3}$, and less than or equal to 1.*

Proof. For a sub-clique Q_s of Q, the edge density of $G[Q_s]$ is clearly 1. Now suppose a near-maximal clique Q is the union of maximal cliques Q' and Q'' with $|Q'| = p$ and $|Q''| = q$, and $p \leq q$. Then all possible numbers of edges in Q is $\binom{q+1}{2}$ and the number of edges in Q is $(p-1) + \binom{q}{2}$. The edge weight of $G[Q]$ is

$$\begin{aligned}
\frac{(p-1) + \binom{q}{2}}{\binom{q+1}{2}} &= \frac{2(p-1) + q(q-1)}{(q+1)q} \\
&\geq \frac{2(q-1) + q(q-1)}{(q+1)q} \\
&= \frac{q^2 + q - 2}{q^2 + q} = 1 - \frac{2}{q^2 + q} \geq \frac{2}{3}
\end{aligned}$$

□

The execution time of the algorithms based on a graph depends on the number of vertices of the graph. For each maximal clique Q, at least $|Q|$ near-maximal cliques can be generated and it might cause increased time complexity since the intersection graph constructed in the next procedure takes all the dense sets we found as vertices. Therefore, we establish rules about which near-maximal cliques must

be included in the pool of dense sets so that we take only selected near-maximal cliques as the dense sets instead of all near-maximal cliques

Rule 1. For a maximal clique $Q = \{e_1, ..., e_k\}$, we take a sub-clique $Q - \{e_i\}$ as a dense-set if a vertex $e_i \in Q$ satisfies the property that the mean of weights on the edges in $L(H)[Q - \{e_i\}]$ is maximized among $e_1, ..., e_k$.

Rule 2. For a pair of maximal cliques Q' and Q'' satisfying $Q' \cap Q''| = |Q'| - 1$, we take the union of these two maximal cliques as a dense set if the mean of weights on the edges joining the vertex in $Q' - Q''$ and the vertices in $Q' \cap Q''$ is greater than or equal to the mean of weights of edges in the maximal clique Q''.

In this work, we compose a family of dense sets in $L'(H)$ using two different algorithms STRONG-CLQ and RELAXED-CLQ.

The algorithm STRONG-CLQ takes the following entities as dense sets. We call vertex sets of this kind "strong dense sets"

(1) maximal-cliques

(2) near-maximal cliques obtained according to **Rule 1** and **Rule 2**

To find more "relaxed" dense sets than the dense sets based on the near-maximal cliques, we merge a number of maximal cliques of which all pairs are max-clique adjacent. We construct a graph whose vertex set is the set of all maximal cliques and two vertices are adjacent if corresponding maximal cliques are max-clique adjacent. Then we find all cliques of the constructed graph using the algorithm of Tsukiyama et al. [TIAS77], denoted by CLIQUES, implemented in the R igraph package [CN06].

The algorithm RELAXED-CLQ takes the following entities as a dense-set. We call vertex sets of this kind "relaxed dense sets"

(1) maximal cliques

(2) sub-cliques (near-maximal cliques) obtained by **Rule 1**

(3) the union of maximal cliques that are pairwise max-clique adjacent

We used the modified Bron–Kerbosh algorithm [ELS10] called MAXCLIQUES to find all maximal cliques of a given graph. This algorithm was implemented in the R igraph package [CN06]. The algorithm to find sub-cliques satisfying Rule 1 is denoted as SUB-CLIQUES, and the algorithm to find all the pairs of max-clique adjacent maximal cliques is named ADJCLIQUES. The three algorithm MAXCLIQUES, SUB-CLIQUES, and ADJCLIQUES are utilized in both algorithms STRONG-CLQ and RELAXED-CLQ. For convenience, we define sub-functions EDGEWEIGHTIN(Q) and EDGEWEIGHTBW(A, B) that calculate the mean of weights of the edges in a maximal clique Q in a given graph, and the mean of weights on the edges between two vertex sets A and B in a given graph, respectively.

Observation 2. *For maximal cliques of a graph G,* Q_1*, ...,* Q_n *such that* $|Q_1| \geq ... \geq |Q_n|$*, if every pair of maximal cliques is max-clique adjacent, then,*

$Q_1 \cap Q_2 = Q_2 - \{v_2\}$ *and* $Q_2 \setminus Q_1 = \{v_2\}$ *for some* $v_2 \in Q_2$

$Q_1 \cap Q_2 \cap Q_3 = Q_3 - \{v_3\}$ *and* $Q_3 \setminus (Q_1 \cup Q_2) = \{v_3\}$ *for some* $v_3 \in Q_3$

$\vdots$

$\bigcap_{l=1}^{n} Q_l = Q_n - \{v_n\}$ *and* $Q_n \setminus \bigcup_{l=1}^{n-1} Q_l = \{v_n\}$

Proof. By the induction hypothesis, $\bigcap_{l=1}^{k} Q_l = Q_k \setminus \{v_k\}$ for some $v_k \in Q_k$. Thus $Q_{k+1} \setminus \bigcap_{l=1}^{k} Q_l = Q_{k+1} \setminus (Q_k - \{v_k\}) = Q_{k+1} \setminus Q_k$. Since Q_k and Q_{k+1} are max-clique adjacent, $Q_{k+1} \setminus Q_k = \{v_{k+1}\}$ for some $v_{k+1} \in Q_{k+1}$. Since $Q_{k+1} \setminus Q_k = \{v_{k+1}\}$, $Q_{k+1} = (Q_{k+1} \cap Q_k) \cup \{v_{k+1}\}$. Now $\bigcap_{l=1}^{k+1} Q_l = Q_{k+1} \cap \bigcap_{l=1}^{k} Q_l = Q_{k+1} \cap (Q_k - \{v_k\}) = Q_{k+1} - \{v_{k+1}\}$ □

Algorithm 5.2: SUB-CLIQUES : Find sub-cliques of each maximal clique satisfying Rule 1

input : A set of all maximal cliques T in $L(H)$ obtained by MAXCLIQUES
output: A list of all sub-cliques L

```
1 L ← ∅;
2 for Q ∈ T do
3     s ← |Q|;
4     for e ∈ Q do
5         w_e ← EDGEWEIGHTBW({e}, Q − {e})
6     if w_e = min{w_e | e ∈ Q} then
7         add the set Q − {e} to a list L
```

Algorithm 5.3: ADJCLIQUES : Find all pairs of maximal cliques that are max-clique adjacent

input : A set of all maximal cliques T in $L(H)$
output: A list M of pairs of maximal cliques that are max-clique adjacent

```
1 M ← ∅;;
2 for i ∈ {1, ..., t} do
3     for j ∈ {1, ..., t} − {i} do
4         if |Q_i ∩ Q_j| = |Q_j| − 1 then
5             M ← {Q_i, Q_j}
```

Algorithm 5.4: STRONG-CLQ : Find strong dense sets in a given graph

input : An $m \times m$ adjacent matrix A of a line graph $L(H)$
output: A family SD of strong dense sets of vertices in $L(H)$

for $i \in \{1, ..., m\}$ **do**
  for $j \in \{j + 1, ..., m\}$ **do**
  if $A[i, j] < \theta$ **then**
    $A[i, j] \leftarrow 0$;
    $A[j, i] \leftarrow 0$;
$CL \leftarrow$ MAXCLIQUES(A);
add all maximal cliques in CL to SD;
add SUB-CLIQUES(CL) to SD;
$AC \leftarrow$ ADJCLIQUES;
for $(Q_i, Q_j) \in AC$ **do**
  $w_i \leftarrow$ EDGEWEIGHTIN(Q_i);
  $w_j \leftarrow$ EDGEWEIGHTBW$(Q_j - Q_i, Q_j \cap Q_i)$;
  if $w_i < w_j$ **then**
    add $Q_i \cup Q_j$ to SD;

Algorithm 5.5: RELAXED-CLQ : Find relaxed dense sets in a given graph

input : An $m \times m$ adjacent matrix A of a line graph $L(H)$
output: A family RD of relaxed dense set of vertices in $L(H)$
/* CLIQUES(AC, k) : Find all cliques size of at least k of a graph whose edge list is AC */

for $i \in \{1, ..., m\}$ **do**
  for $j \in \{j + 1, ..., m\}$ **do**
  if $A[i, j] < \theta$ **then**
    $A[i, j] \leftarrow 0$;
    $A[j, i] \leftarrow 0$;
$CL \leftarrow$ MAXCLIQUES(A);
add all maximal cliques in CL to RD;
add SUB-CLIQUES(CL) to RD;
$AC \leftarrow$ ADJCLIQUES;
add CLIQUES$(AC, 2)$ to RD;

5.3.4 Finding the MWIS of the intersection graph

In the previous procedure, we listed the dense sets of the line graph $L'(H)$. Let $\mathcal{Q} = \{Q_1, ..., Q_k\}$ be a family of dense sets in $L'(H)$. The union of hyperedges corresponding to each dense set can be regarded as a candidate cluster of the hypergraph, and we call these sets "candidate vertex sets", abbreviated to "Csets". For the family of dense sets $\mathcal{Q} = \{Q_1, ..., Q_k\}$, we can find a family of Csets, $\mathcal{C} = \{C_1, ..., C_k\}$ where $C_i = \bigcup_{e \in Q_i} e$. With the family of Csets $\mathcal{C}$ and the family of dense sets Q, we can construct intersection graphs $I_{\mathcal{C}}$ and $I_{\mathcal{Q}}$, respectively. We assign the same weight $\frac{\sum_{e \in Q_i} w'_H(e)}{|C_i|}$ to Q_i and C_i as the vertex weight for $I_{\mathcal{Q}}$ and $I_{\mathcal{C}}$, respectively.

To find the MWIS of the intersection graph $I_{\mathcal{C}}$ or $I_{\mathcal{Q}}$, we use the greedy algorithm GWMIN suggested by Sakai et al. [STY03]. We can determine final clusters using the MWIS of both intersection graphs $I_{\mathcal{C}}$ and $I_{\mathcal{Q}}$, yet there is some difference in the details of the procedures.

Before we introduce the details of the remaining procedures of our algorithms, we first observe the difference between the two algorithms based on $I_{\mathcal{C}}$ and $I_{\mathcal{Q}}$. Note that when two dense sets Q_i and Q_j are not adjacent in $I_{\mathcal{Q}}$, the two corresponding Csets C_i and C_j might share vertices of $V(H)$, and hence the edge-density of $I_{\mathcal{C}}$ is greater than or equal to that of $I_{\mathcal{C}}$. When we choose a vertex of the intersection graph during the procedure of GWMIN, more vertices might be removed from the intersection graph in the case of $I_{\mathcal{C}}$. Therefore, the MWIS of $I_{\mathcal{C}}$ might be much smaller than that of $I_{\mathcal{Q}}$ and, as a result, many vertices might remain after we take clusters by choosing the Csets in MWIS of $I_{\mathcal{C}}$. To avoid this problem, we apply a recursive procedure for the algorithm using $I_{\mathcal{C}}$.

Clustering on $I_{\mathcal{C}}$. We find the MWIS of the intersection graph $I_{\mathcal{C}}$ using GWMIN.

Each Cset in MWIS is taken as a cluster. Then we update the hyperedges of the original hypergraph by removing the vertices that belong to taken clusters. With these updated hyperedges, we repeat the whole procedure again until every vertex of the hypergraph is assigned to a cluster or remains as a singleton. For the remaining singletons, we can assign them to the already constructed clusters. The singleton assignment rule (SGTAPPEND) is as follows.

Let $\Pi = \{P_1, ..., P_k\}$ be a set of found clusters, and let $s \in V(H)$ be the singleton which do not belong to any $P_i \in \Pi$. We calculate $p_i = \max_{s \in e}\{|P_i \cap e|\}$ for each P_i, and assign the singleton v to the cluster with $\max\{p_1, \cdots, p_k\}$. If there are more than two clusters $P_{x_1}, \cdots, P_{x_t}$ with $\max\{p_1, \cdots, p_k\}$, we calculate $q_{x_i} = \sum_{s \in e} |P_{x_i} \cap e|$ for each P_{x_i} and assign the singleton v to the cluster with $\max\{q_{x_1}, \cdots, q_{x_t}\}$.

Clustering on $I_{\mathcal{Q}}$. We find the MWIS of the intersection graph $I_{\mathcal{Q}}$ using GWMIN and we take Csets that correspond to the dense sets in the MWIS. However, as we observed, for the two vertices Q_i and Q_j in MWIS, corresponding sets C_i and C_j of $\mathcal{C}$ might intersect. To find distinct partitions, we order the MWIS from maximum weights to minimum weights and then take corresponding sets in $\mathcal{C}$ as clusters according to that order, except already taken vertices. For the remaining vertices that are not taken as a member of a cluster, we can assign the vertices to a proper cluster by applying the previous singleton assignment rule.

For convenience, we call the algorithms used to find clusters by detecting the MWIS of intersection graph $I_{\mathcal{C}}$ and $I_{\mathcal{Q}}$ CSET-CLST and QSET-CLST, respectively. In summary, we can apply one of four different algorithms to partition a sparse hypergraph.

- PSHSC: CSET-CLST combined with STRONG-CLQ.

- PSHRC: CSET-CLST combined with RELAXED-CLQ.
- PSHSQ: QSET-CLST combined with STRONG-CLQ.
- PSHRQ: QSET-CLST combined with RELAXED-CLQ.

Theorem 2. *The algorithms* PSHSC*,* PSHRC*,* PSHSQ*, and* PSHRQ *are all convergent.*

Given a hypergraph H, when we construct an intersection graph of the family of candidate clusters, the size of the MWIS of the intersection graph is at least 1. For the repetitive algorithms PSHSC and PSHRC, at least one or more of vertices in $V(H)$ must be taken as cluster members in each iteration. Thus, the size of the vertex set of a hypergraph is monotone decreasing after each iteration and, therefore, the algorithms PSHSC and PSHRC are convergent. For the algorithms PSHSQ and PSHRQ, there must be at least one cluster detected by MWIS, and hence they are convergent.

Algorithm 5.6: GWMIN [STY03] : Greedy algorithm to find MWIS

input : A vertex weighted graph $I_{\mathcal{F}}$
output: A maximum weight independent set M in $I_{\mathcal{F}}$

1 $M \leftarrow \emptyset$; $i \leftarrow 0$; $I_i \leftarrow I_{\mathcal{F}}$;
2 **while** $V(I_i) \neq \emptyset$ **do**
3 Choose a vertex, say v_i, maximizes the function $w(u)/[d_{I_i}(u) + 1]$ in I_i;
4 $M \leftarrow I \cup \{v_i\}$; $I_i \leftarrow G_i[V(I_i) - N^+_{I_i}(v_i)]$;
5 $i \leftarrow i + 1$;

Algorithm 5.7: CSET-CLST

input : A hypergraph H, boolen $appendSGT$
output: A list of clusters $Final$

$Vs \leftarrow V(H)$;
$Hs \leftarrow E(H)$;
$Final \leftarrow \emptyset$;
while $|Hs| \neq 0$ **do**
  $A \leftarrow$ ADJLH(Hs);
  if $A = 0$ **then**
    break;
  $\mathcal{Q} \leftarrow$ STRONG-CLQ(A)
  (or $\mathcal{Q} \leftarrow$ RELAXED-CLQ(A));
  $\mathcal{C} \leftarrow \emptyset$;
  for $Q_i \in \mathcal{Q}$ **do**
    $C_i \leftarrow \bigcup_{e \in Q_i} e$;
    add C_i to $\mathcal{C}$;
  $A \leftarrow k \times k$ zero matrix (adjacency matrix of $I_{\mathcal{C}}$);
  for $i \in \{1, ..., k-1\}$ **do**
    for $j \in \{i+1, ..., k-1\}$ **do**
      if $|C_i \cap C_j| \neq \emptyset$ **then**
        $A[i,j] \leftarrow 1$; $A[j,i] \leftarrow 1$;
  for $C_i \in \mathcal{C}$ **do**
    $w_{V(I_{\mathcal{C}})}(C_i) \leftarrow \sum_{e_{i_k}, e_{i_l} \in Q_i} A[i_k, i_l]/|C_i|$;
  $M \leftarrow$ GWMIN$(I_{\mathcal{C}})$ add M to $Final$;
  $Vs \leftarrow Vs - \bigcup S \in MS$;
  for $e \in Hs$ **do**
    $e \leftarrow e \cap Vs$
  remove all empty hyperedges in Hs from Hs;
$SGT \leftarrow Vs$;
if $SGT \neq \emptyset$ **then**
  if $appendSGT = true$ **then**
    for $v \in SGT$ **do**
      SGTAPPEND(v);
else
  add SGT to $Final$

Algorithm 5.8: QSET-CLST

input : A hypergraph H, boolen $appendSGT$
output: A list of clusters $Final$

$Vs \leftarrow V(H)$; $Hs \leftarrow E(H)$;
$Final \leftarrow \emptyset$;
$A \leftarrow$ ADJLH(Hs);
if $A = 0$ **then**
    break;
$\mathcal{Q} \leftarrow$ STRONG-CLQ(A) (or $\mathcal{Q}$RELAXED-CLQ(A));
$\mathcal{C} \leftarrow \emptyset$;
for $Q_i \in \mathcal{Q}$ **do**
    $C_i \leftarrow \bigcup_{e \in Q_i} e$;
    add C_i to $\mathcal{C}$;
$A \leftarrow k \times k$ zero matrix (adjacency matrix of $I_{\mathcal{Q}}$);
for $i \in \{1, ..., k-1\}$ **do**
    for $j \in \{i+1, ..., k-1\}$ **do**
        if $|Q_i \cap Q_j| \neq \emptyset$ **then**
            $A[i,j] \leftarrow 1$; $A[j,i] \leftarrow 1$;
for $Q_i \in \mathcal{Q}$ **do**
    $w_{V(I_{\mathcal{Q}})}(Q_i) \leftarrow \sum_{e_{i_k}, e_{i_l} \in Q_i} A[i_k, i_l]/|C_i|$;
$M \leftarrow$ GWMIN$(I_{\mathcal{Q}})$;
while $M \neq \emptyset$ **do**
    choose $Q \in M$ if $w(Q) = max\{w(Q') | Q' \in M\}$;
    if $\bigcup_{e \in Q} e \cap Vs \neq \emptyset$ **then**
        add $\bigcup_{e \in Q} e \cap Vs$ to $Final$
    else
        remove Q from M
    $Vs \leftarrow Vs - \bigcup Final$
$SGT \leftarrow Vs$;
if $SGT \neq \emptyset$ **then**
    if $appendSGT = true$ **then**
        for $v \in SGT$ **do**
            SGTAPPEND(v);
else
    add SGT to $Final$

5.4 Experiments

5.4.1 Simulations

Simulating the hypergraph with cluster structure

We generate various hypergraphs that have an underlying cluster structure. For a hypergraph that has an underlying cluster structure, we can consider three types of hyperedges: a hyperedge that fully belongs to a cluster (type 1); a hyperedge that does not fully belong to, but is certainly associated with a specific cluster (type 2); and a hyperedge that is not associated with any cluster (type 3). For this, we use six different parameters for the number of vertices (n), the number of hyperedges (m), the number of hypergraphs (k), the ratio of the number of hyperedges of type 1 and 2 over the number of all hyperedges (p), the ratio of the number of hyperedges of type 1 over the number of hyperedges type 1 and 2 (q), and the probability of each vertex of a hyperedge of type 2 being chosen among the vertices belonging to the associated cluster (r). We generate 100 hypergraphs for each n=100, 200, 500, 1000, 2000, 5000, and 10000. For each n, the number of hyperedges m is randomly chosen between $0.2n$ and n to construct sparse hypergraphs. The parameter k is randomly selected between $\max(0.02m, 2)$ and $\min(0.2m, 0.1n)$. Each parameter, p, q, and r, is randomly chosen between 0.85 and 0.99. The detailed procedure to generate a simulated hypergraph is as follows.

STEP 1. For a fixed number n, randomly select the value of other parameters m, k, p, q, r according to the above conditions.

STEP 2. Partition $1, ..., n$ into k parts $A_1, ..., A_k$ such that the size of each part is between $0.5n/k$ and $1.5n/k$.

STEP 3. Let E_1, E_2, and E_3 be the sets of hyperedges of type 1, 2, and 3 respectively. Then $|E_1 \cup E_2| = np$ and $E_3 = n - np$. For each cluster A_i, randomly

choose a number a_i of hyperedges associated with the cluster A_i such that $|a_i|$ must be at least $\lceil 0.8 \times |E_1 \cup E_2|/k \rceil$

STEP 4. To construct hyperedges that are associated with a cluster A_i, we compose $\lceil a_i q + 0.5 \rceil$ hyperedges of type 1 and $a_i - \lceil a_i q + 0.5 \rceil$ hyperedges of type 2. Randomly choose vertices of each hyperedge of type 1 among the vertices in a cluster A_i such that the size of the hyperedge must be between 3 and $|A_i|$. For a hyperedge of type 2, choose the size of the hyperedges between 3 and A_i and select each vertex from A_i and $\{1, ..., n\} \setminus A_i$ with probabilities of r and $1 - r$, respectively.

STEP 5. To compose hyperedges of type 3, randomly decide the size of each hyperedge between 3 and $\max\{|A_1|, ..., |A_k|\}$ and randomly select the vertices from $1, ..., n$ as the number of decided size.

Execution of algorithms for simulated data

We obtained the results of the clustering algorithms PSHSC, PSHRC, PSHSQ, and PSHRQ for simulated data. We also obtained the results of the hMETIS program [KK98a, KK98b] as a multi-level hypergraph clustering for comparisons. To run algorithms PSHSC, PSHRC, PSHSQ, and PSHRQ, we set the threshold $\theta = 0.5$ to delete edges with low weights. In addition, we utilized the rule to assign singletons to already constructed cluster as stated in section 5.3.4. In the hMETIS program package, we run the shmetis program because it is the program in which the options for coarsening, initial partitioning, and uncoarsening phase are already set as options that perform reasonably well for all types of experimental data. We only need to set the imbalance parameter $UBfactor$ and the desired number of parts $Npart$ for shmetis. For a bisection, the size of each part will be between $\frac{50-UBfactor}{100} \times n$ and $\frac{50+UBfactor}{100} \times n$, and for 2^k-way partitioning, each part can contain $(\frac{50-UBfactor}{100})^k \times n$ to $(\frac{50+UBfactor}{100})^k \times n$. We set $UBfactor = 20$ for

stable execution. Since we already know the true number of the underlying clusters, k, we set $Npart = 0.5k$, k, $1.5k$, and $2k$ (the results are denoted as hMETIS0.5K, hMETISK, hMETIS1.5K, and hMETIS2K, respectively).

5.4.2 Quality measures

We evaluate the performance of the clustering algorithms in terms of the density of the detected clusters and the agreement of the resulting clusters with the underlying cluster structure. We used the *fitness measure*, *connectivity measure*, *conductance measure*, *hyperedge cut*, *scaled cost*, *absorption*, *partitioning cost*, and *connectivity measure*, to show how similar the vertices in a cluster are.

First, we introduce some notions about partitioning of a hypergraph.

Let $\Pi = \{P_1, P_2, ..., P_k\}$ be a partition of a hypergraph H. A vertex $v \in V$ is said to be *assigned* to a part π if $v \in \pi$. A hyperedge $e \in E$ is said to be *associated* with the part $\pi \in \Pi$ if $e \cap \pi \neq \emptyset$. The *connectivity degree* of a hyperedge e is the number of parts connected to e and is denoted by $\lambda_e(H, \Pi)$. A hyperedge is said to be a *cut* if its connectivity degree is more than 1.

The fitness measure [HKKM98], the conductance measure [LLM10], and the connectivity measure [HKKM98] are local measures that show the quality of each partition (or its vertices,) we calculate the measures of each cluster of obtained partitioning results, and then average the values.

A. The *fitness measure* [HKKM98] is the ratio of weights of edges that are within the cluster and weights of edges that are connected to the cluster. The fitness function that measures the goodness of a cluster S is defined as

$$fitness(S) = \frac{\sum_{e \subset S} w'_H(e)}{\sum_{|e \cap S| > 0} w'_H(e)}$$

In our experiments, we set the $w'_H(e) = |e|$.

B. The *conductance measure* [LLM10] is the ratio between the number of edges inside the cluster and the number of edges leaving the cluster. The conductance $\phi(S)$ of a cluster S is $\phi(S) = c_s/\min(Vol(S), Vol(V \backslash S))$, where c_S is the number of external edges, and $Vol(S) = \sum_{u \in S} d_H(u)$, where $d_H(u)$ is the degree of vertex u.

C. The *connectivity measure* [HKKM98] is the percentage of edges inside the cluster that each vertex is associated with. The connectivity function of vertex v in cluster S is defined as

$$connectivity(v, S) = \frac{|\{e|e \subset S, v \in e\}|}{|\{e|e \subset S\}|}$$

The vertices with high connectivity measure can be considered to belong to the cluster. For a given hypergraph clustering result, we calculate the connectivity measure for each vertex and the part associated with the vertex, and then calculate the mean of all the obtained connectivity measures. If $|\{e|e \subset S\}| = 0$ for a cluster, we except the measure of the cluster when calculating the mean of the connectivity measure.

The partitioning cost, the hyperedge cut, the scaled cost [CSZ94], and the absorption [SS95] are the global measures that show the quality of the whole clustering results.

D. Given a hypergraph H with a partition Π, we can define the *partitioning cost* of Π as

$$C(H, \Pi) = \sum_{e \in E} (w'_H(e) \cdot (\lambda_e(H, \Pi) - 1))$$

E. The *hyperedge cut* is the number of the hyperedges connected to multiple partitions. The hMETIS algorithms are designed to try to minimize the hyperedge cut of a hypergraph.

F. The *scaled cost* [CSZ94] is defined as

$$\frac{1}{n(k-1)} \sum_{i=1}^{k} \frac{|\{e|e \cap P_i \neq \emptyset, \text{and } e \nsubseteq P_i\}|}{|P_i|}$$

where $\Pi = \{P_1, ..., P_k\}$ is the partition of a hypergraph. The low scaled cost guarantees the lower cut and more balanced partition sizes, and also the scaled cost is optimized for the partitions in which the smaller-sized partitions have smaller edge cuts [CSZ94].

G. The *Absorption* [SS95] is defined as

$$\sum_{i=1}^{k} \sum_{e \in E(H), e \cap P_i \neq \emptyset} \frac{|e \cap P_i| - 1}{|e| - 1}$$

where $\Pi = \{P_1, ..., P_k\}$ is the partition of a hypergraph.

The Rand index and the adjusted Rand index are the cluster validation measures used to show how well the clustering results represent the inherent cluster structure of the data.

H. The *Rand index* [Ran71] is the measure of the similarity between two clustering results. In this study, we generate the simulated hypergraph that contains an underlying clustering structure. By comparing the clustering results with underlying clustering structure using the Rand index, we can evaluate the performance of the suggested algorithm. If we let $a = |\{(v_i, v_j)|v_i, v_j \in X_k, v_i, v_j \in Y_l\}|$ and $b = |\{(v_i, v_j)|v_i \in X_{k_1}, v_j \in X_{k_2}, v_i \in Y_{l_1}, v_j \in Y_{l_2}\}|$ for for two clustering results $X = \{X_1, X_2, ..., X_p\}$ and $Y = \{Y_1, Y_2, ..., Y_q\}$, then the Rand index RI is

$$RI = \frac{a+b}{\binom{n}{2}}.$$

I. The adjusted form of the Rand Index, the *Adjusted Rand Index* [HA85], is defined as $ARI = \frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex}$, i.e.,

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}$$

where $n_{ij} = |X_i \cap Y_j|$, $a_i = \sum_i n_{i\cdot}$, and $b_j = \sum_j n_{\cdot j}$. Then the adjusted Rand index is expected to be zero for independent clusters and can be at most 1.

J. For each clustering result for a simulated hypergraph, we calculate the difference between the number of clusters of partitioning results and the number of true underlying clusters k, denoted by $|k - \text{N.clst}|$.

5.4.3 Results

We implemented our algorithms (PSHSC, PSHRC, PSHSQ, and PSHRQ) in R [R C17]. In order to speed up the execution of algorithms, we used R library package *Rcpp* [EF11, Edd13] to integrate R and C++. For the modified Bron–Kerbosch algorithm [BK73, ELS10] that finds all maximal cliques in a graph, required in RELAXED-CLQ and STRONG-CLQ implementation, we used the R library *igraph* package [CN06]. For the algorithm to find all cliques in the graph, required in RELAXED-CLQ, we used the method of Tsukiyama et al. [TIAS77] implemented in R library *igraph* package. We obtained the clustering results of the simulation hypergraphs using the algorithms and the program hMETIS [KK98a, KK98b]. For hMETIS, we used four different $Npart$ parameters (hMETIS0.5K, hMETISK, hMETIS1.5K, and hMETIS2K). We calculated the local quality measures (fitness, conductance, and connectivity), global quality measures (scaled cost, absorption, partitioning cost, and hyperedge cut), and validation measures (Rand index, adjusted Rand index, and $|k - \text{N.clst}|$).

In Table 5.1, we summarize the mean of every quality measure for each method and n. The high fitness measure of a part shows that the partition has more weight for the edges within a part. The fitness values of hMETIS0.5K are highest for every n, and the values of PSHRQ are next highest except for the case of $n = 200$. Most of the fitness values of hMETIS are higher than those of PSHSC and PSHRC, but lower than those of PSHSQ and PSHRQ. The fitness values tend to decrease as n increases. The lower conductance measure of a part means that the number of edges associated with the part but not fully contained in the part is relatively small compared with the number of edges fully contained in the part. For the entire range of n, the conductance measures of hMETIS0.5K are lowest. The conductance values of hMETISK are lower than the values of PSHSQ and PSHRQ for $n = 100$. However, as n increases, the conductance values of PSHSQ and PSHRQ are lower than hMETISK especially for $n \geq 1000$. The conductance values of PSHSC and PSHRC are lower than hMETIS2K and hMETIS1.5K in general. The high connectivity measure of a vertex in a part captures that a vertex of a cluster is better connected with the vertices in the part than outside the part. The connectivity values of PSHSC and PSHRC are highest for the entire range of n, and that of PSHSQ and PSHRQ are next highest. On the other hand, the connectivity values of hMETIS0.5K are lowest while the fitness and conductance values are better than with the other methods.

The lower scaled cost means that the partitions have smaller edge cuts and balanced sizes. For $n \leq 1000$, the scaled cost values of hMETIS0.5K are significantly lower than the other methods, but for large $n \geq 2000$, the metrics of all methods are similar. The absorption measure shows the sum of the ratio of the number of vertices in both an edge and a part over the number of vertices in an edge, and absorption values for all methods are similar in the entire range of n. The partitioning cost is

a measure of the weight and number of edges that are divided by the partitions, and the hyperedge cut is the measure of the number of edges that are split by the partitions. The partitioning cost values of hMETIS0.5K and hMETISK are lower than those of our algorithms. The hyperedge cut values of hMETIS0.5K is lowest, and hyperedge cut values of PSHSC and PSHRC are next lowest, and those of PSHSQ and PSHRQ are also lower than hMETISK for $n \geq 2000$, while the hyperedge cut values of hMETISK are lower than those of our four algorithms for $n \leq 200$.

The higher Rand index and adjusted Rand index show the agreement of a partitioning result with the true underlying cluster structure. The four algorithms PSHSC, PSHRC, PSHSQ, and PSHRQ have similar values for the Rand index and adjusted Rand index, but hMETIS methods show significantly lower values than those of our algorithms. It is notable that the Rand index and adjusted Rand index measures of hMETIS0.5K are lowest, and even worse than those of hMETIS2K. Since our algorithms do not specify the number of final clusters, the differences in the number of parts between underlying clusters and results of our algorithms are higher than that between underlying clusters and results of hMETISK. However, the number of detected clusters using our algorithms, especially for PSHSQ and PSHRQ, remained in a reasonable range even for a large n. For $n = 10000$, the mean of the number of underlying clusters (k) was 541, while the mean of $|k - \text{N.clst}|$ was 43.6 and 38.4 for PSHSQ and PSHRQ, respectively.

The results show that every quality measure of PSHRC is slightly better than PSHSC, and that every quality measure of PSHSQ is slightly better than PSHRQ. The algorithms PSHSQ and PSHRQ that implement QSET-CLST show significantly higher quality metrics than the algorithms PSHRC and PSHSC that implement CSET-CLST in terms of fitness, conductance, scaled cost, partitioning cost, and $|k - \text{N.clst}|$. On the other hand, PSHRC and PSHSC showed better qual-

Table 5.1: Summary of quality measures for each method and the number of vertices in a hypergraph (n)

n	method	fitness	conductance	connectivity	scaledcost	absorption	paritioning cost	hyperedge cut	Rand	adjusted Rand	$\lvert k - \text{N.clst}\rvert$
100	PSHSC	0.515	0.164	0.534	0.0074	55.13	23.3	10.3	0.979	0.939	1.9
100	PSHRC	0.533	0.150	0.533	0.0070	55.21	22.7	9.9	0.981	0.945	1.7
100	PSHSQ	0.577	0.070	0.530	0.0042	55.49	19.6	9.7	0.985	0.955	0.9
100	PSHRQ	0.580	0.066	0.530	0.0042	55.50	19.7	9.8	0.985	0.956	0.8
100	hMETISK	0.557	0.059	0.470	0.0047	55.92	18.5	9.6	0.953	0.865	0.0
100	hMETIS0.5K	0.759	0.023	0.313	0.0022	57.31	8.0	4.9	0.814	0.565	2.5
100	hMETIS1.5K	0.273	0.136	0.400	0.0095	54.24	38.2	22.5	0.915	0.760	2.5
100	hMETIS2K	0.123	0.221	0.336	0.0145	52.39	59.4	33.6	0.879	0.641	5.1
200	PSHSC	0.395	0.180	0.521	0.0055	111.13	78.7	24.5	0.983	0.935	3.5
200	PSHRC	0.404	0.167	0.520	0.0052	111.31	78.3	23.6	0.983	0.938	3.2
200	PSHSQ	0.441	0.087	0.509	0.0028	112.15	65.5	25.0	0.988	0.954	1.2
200	PSHRQ	0.434	0.086	0.503	0.0028	112.13	66.6	25.7	0.987	0.952	1.1
200	hMETISK	0.445	0.089	0.448	0.0030	113.24	59.6	22.3	0.966	0.848	0.0
200	hMETIS0.5K	0.657	0.037	0.301	0.0015	119.26	31.2	11.2	0.878	0.576	5.3
200	hMETIS1.5K	0.219	0.181	0.399	0.0057	109.70	99.5	46.7	0.952	0.794	5.3
200	hMETIS2K	0.109	0.273	0.350	0.0083	106.24	139.3	65.1	0.935	0.714	10.6
500	PSHSC	0.347	0.204	0.524	0.0026	266.50	268.4	53.8	0.994	0.950	8.5
500	PSHRC	0.356	0.189	0.524	0.0024	267.18	263.9	52.3	0.994	0.953	7.5
500	PSHSQ	0.366	0.105	0.505	0.0014	268.40	245.6	62.4	0.996	0.965	2.9
500	PSHRQ	0.370	0.100	0.503	0.0014	268.51	246.2	62.7	0.996	0.965	2.4
500	hMETISK	0.351	0.110	0.415	0.0016	271.01	225.4	59.7	0.978	0.816	0.0
500	hMETIS0.5K	0.541	0.052	0.285	0.0008	277.62	134.5	28.3	0.922	0.569	11.6
500	hMETIS1.5K	0.178	0.198	0.386	0.0027	263.21	327.7	113.5	0.979	0.818	11.8
500	hMETIS2K	0.095	0.262	0.340	0.0037	255.32	424.0	154.5	0.972	0.749	23.4
1000	PSHSC	0.285	0.235	0.527	0.0018	568.97	781.8	125.1	0.996	0.954	16.1
1000	PSHRC	0.293	0.217	0.526	0.0017	570.81	771.7	122.2	0.996	0.957	13.9
1000	PSHSQ	0.298	0.125	0.491	0.0010	573.19	737.8	152.6	0.996	0.960	3.7
1000	PSHRQ	0.299	0.122	0.488	0.0010	573.00	743.7	155.5	0.996	0.959	3.1
1000	hMETISK	0.269	0.148	0.389	0.0011	579.19	661.9	144.5	0.983	0.760	0.0
1000	hMETIS0.5K	0.422	0.078	0.276	0.0006	593.96	459.7	72.4	0.954	0.540	24.7
1000	hMETIS1.5K	0.146	0.224	0.378	0.0016	560.93	876.4	249.3	0.988	0.785	24.9
1000	hMETIS2K	0.087	0.292	0.355	0.0021	543.63	1084.3	320.9	0.986	0.738	49.3
2000	PSHSC	0.290	0.247	0.542	0.0007	1085.69	1531.2	204.3	0.999	0.960	36.6

2000	PSHRC	0.300	0.233	0.542	0.0007	1088.37	1514.2	194.8	0.999	0.965	32.1
2000	PSHSQ	0.327	0.137	0.524	0.0005	1093.67	1476.8	225.5	0.999	0.970	11.2
2000	PSHRQ	0.330	0.133	0.522	0.0004	1094.15	1480.0	224.8	0.999	0.970	9.8
2000	hMETISK	0.269	0.157	0.394	0.0005	1100.46	1407.9	259.7	0.993	0.752	0.0
2000	hMETIS0.5K	0.406	0.091	0.280	0.0003	1126.50	1042.5	117.9	0.979	0.536	54.6
2000	hMETIS1.5K	0.148	0.238	0.388	0.0008	1066.39	1797.4	456.0	0.995	0.765	54.8
2000	hMETIS2K	0.088	0.305	0.365	0.0010	1030.98	2182.3	604.8	0.994	0.716	108.9
5000	PSHSC	0.250	0.268	0.532	0.0004	2634.82	4501.8	541.9	0.999	0.957	89.7
5000	PSHRC	0.258	0.255	0.532	0.0004	2640.84	4471.6	524.0	0.999	0.960	80.1
5000	PSHSQ	0.283	0.138	0.506	0.0002	2654.26	4336.7	640.8	1.000	0.968	23.6
5000	PSHRQ	0.285	0.135	0.504	0.0002	2655.28	4340.3	640.1	1.000	0.968	20.5
5000	hMETISK	0.233	0.157	0.376	0.0002	2674.42	4059.6	672.7	0.996	0.714	0.0
5000	hMETIS0.5K	0.356	0.096	0.274	0.0002	2730.96	3192.3	304.4	0.989	0.516	128.4
5000	hMETIS1.5K	0.130	0.235	0.365	0.0003	2597.74	4995.4	1126.8	0.997	0.728	128.0
5000	hMETIS2K	0.080	0.297	0.346	0.0004	2514.99	5938.5	1464.1	0.997	0.693	255.1
10000	PSHSC	0.261	0.275	0.534	0.0002	5819.35	9509.6	1220.9	1.000	0.956	183.7
10000	PSHRC	0.269	0.260	0.533	0.0002	5831.21	9450.4	1190.1	1.000	0.959	164.1
10000	PSHSQ	0.293	0.147	0.512	0.0001	5866.36	9086.4	1421.1	1.000	0.968	43.6
10000	PSHRQ	0.294	0.144	0.509	0.0001	5865.48	9133.7	1437.1	1.000	0.967	38.4
10000	hMETISK	0.232	0.175	0.376	0.0001	5898.07	8754.8	1505.3	0.998	0.689	0.0
10000	hMETIS0.5K	0.347	0.110	0.274	0.0001	6018.85	7245.9	707.8	0.994	0.491	270.4
10000	hMETIS1.5K	0.129	0.255	0.366	0.0002	5724.96	10583.5	2522.6	0.998	0.707	268.4
10000	hMETIS2K	0.079	0.321	0.348	0.0002	5537.56	12487.6	3229.5	0.998	0.675	535.8

ity metrics than PSHSQ and PSHRQ in terms of connectivity and hyperedge cut. The algorithms that implement RELAXED-CLQ showed slightly higher quality metrics than the algorithms that implement STRONG-CLQ if the other conditions are identical. In particular, PSHRC shows values that were better than or equal to PSHSC for almost all quality measures except connectivity, and differences in the connectivity are insignificant.

For hMETIS, the results of hMETIS0.5K (hMETIS with $Npart = 0.5k$) show the best quality metrics in terms of fitness, conductance, scaled cost, absorption, partitioning cost, and hyperedge cut, that are better than not only the other hMETIS results but also all the experimental results. However, the connectivity, Rand index, and adjusted Rand index of the hMETIS0.5K reveal the worst metrics for all n. All quality metrics worsen as the $Npart$ increases from k to $2k$.

As n increases, the metrics of fitness, and scaled cost tend to decrease, and the metrics of conductance, absorption, partitioning cost, and hyperedge cut also show a tendency to increase in all methods, which is expected since the number and sizes of edges, and clusters also increase as n increases. Our algorithms automatically decide the number of partitions, and the differences between the true number of clusters and the number of detected clusters tend to increase as the n increases. Meanwhile, the connectivity of partitioning results obtained by our algorithms (PSHSC, PSHRC, PSHSQ, and PSHRQ) do not decrease as much as the values of hMETIS.

In Table 5.2, we summarize the mean of every quality measure for each method. The average metrics of fitness, conductance, scaled cost, partitioning cost, Rand measure, adjusted Rand measure, and $|k - \text{N.clst}|$ for PSHRQ and PSHSQ are better than those for PSHRC and PSHSC, although the average metrics of connectivity and hyperedge cut for PSHRC and PSHSC are slightly better than those

Table 5.2: Summary of every quality measure for each method

method	fitness	conductance	connectivity	scaledcost	absorption	paritioning cost	hyperedge cut	Rand	adjusted Rand	$\lvert k - \text{N.clst}\rvert$
PSHSC	0.335	0.225	0.530	0.003	1505.94	2385.0	311.5	0.993	0.950	48.55
PSHRC	0.345	0.210	0.530	0.003	1509.28	2367.5	302.4	0.993	0.954	43.22
PSHSQ	0.369	0.116	0.511	0.001	1517.64	2281.2	362.4	0.995	0.963	12.43
PSHRQ	0.370	0.112	0.509	0.001	1517.72	2290.1	365.1	0.995	0.963	10.86
hMETISK	0.337	0.128	0.410	0.002	1527.47	2169.7	382.0	0.981	0.778	0.01
hMETIS0.5K	0.493	0.071	0.286	0.001	1597.60	1773.3	182.4	0.935	0.541	72.77
hMETIS1.5K	0.175	0.210	0.383	0.003	1482.45	2674.0	648.2	0.975	0.765	70.81
hMETIS2K	0.094	0.281	0.349	0.004	1434.44	3187.9	838.9	0.966	0.704	141.16

for the PSHRQ and the PSHSQ. Among the two algorithms PSHRQ and PSHSQ, PSHRQ shows slightly better performance than PSHSQ. The number of clusters detected by PSHSQ and PSHRQ are not exactly the same as the number of true underlying clusters compared with the results of hMETIS using $Npart = k$, but the quality measures of the clustering results of those two methods show better performance metrics on average.

Figure 5.1 shows the local quality measures for various edge densities. The fitness values of our algorithms are slightly increased as the edge density increases. The fitness values of PSHRQ and PSHSQ are slightly higher than PSHSC, PSHRC, and hMETISK, and the values of hMETIS2K is highest among all methods. The conductance values of PSHRQ and PSHSQ retain the relatively lower values for all edge densities except hMETIS0.5K, and the values of PSHSC and PSHRC decrease as the edge density increases, but is still higher than for PSHRQ and PSHSQ. When we take $Npart$ as the greater number than the number of true underlying clusters, the conductance values rapidly increase as the edge density increases. The connectivity values of our algorithms are relatively higher than those of all hMETIS methods, and also have a tendency to increase as the edge density increases. The fitness values and conductance values of the hMETI0.5K are better than any other methods, but the connectivity values were quite a lot worse than any other methods, even including hMETIS1.5K and hMETIS2K.

Figure 5.2 shows the global quality measures for the various edge densities. As the edge density increases, all global quality metrics tend to increase. In all edge density ranges,the global quality measures of our four algorithms and hMETISK are quite similar.

The Figure 5.3 shows the validation quality measures for the various edge density values. Undoubtedly, $|k - \text{N.clst}|$ is lowest for hMETISK, yet we can see that

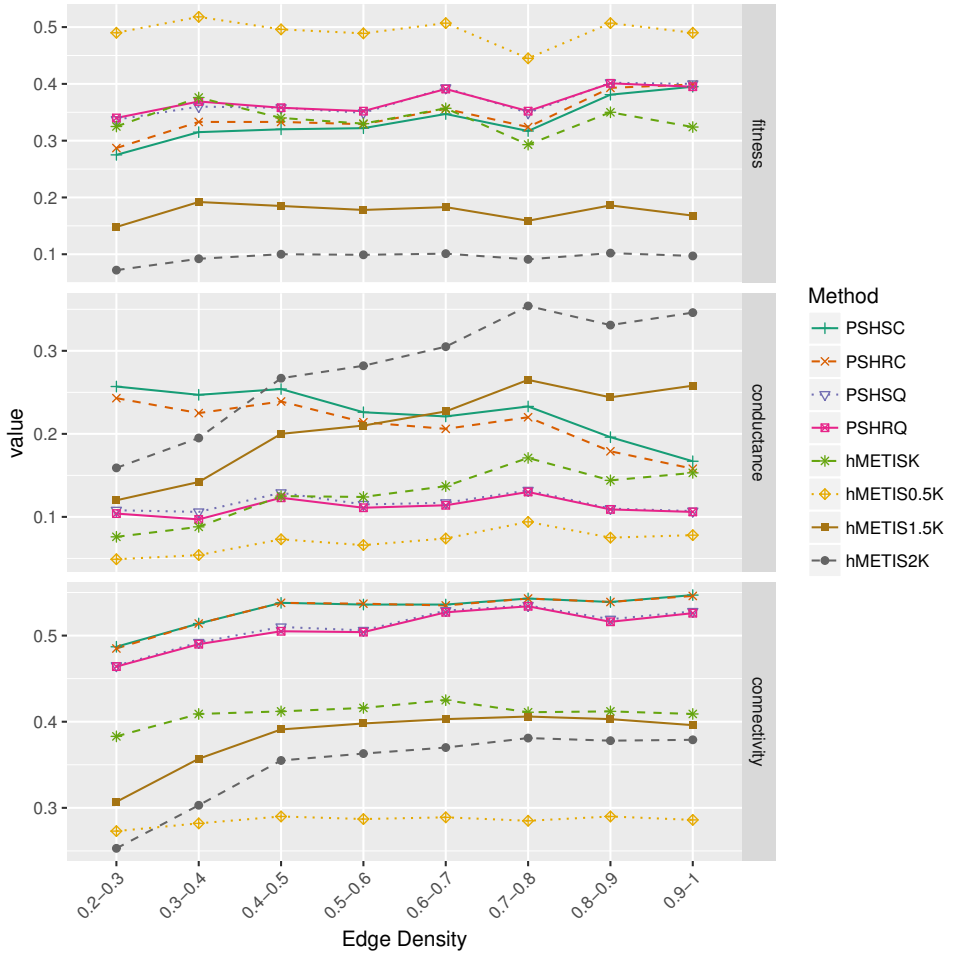


Figure 5.1: Local quality measures (fitness, conductance, and connectivity) for the various edge densities

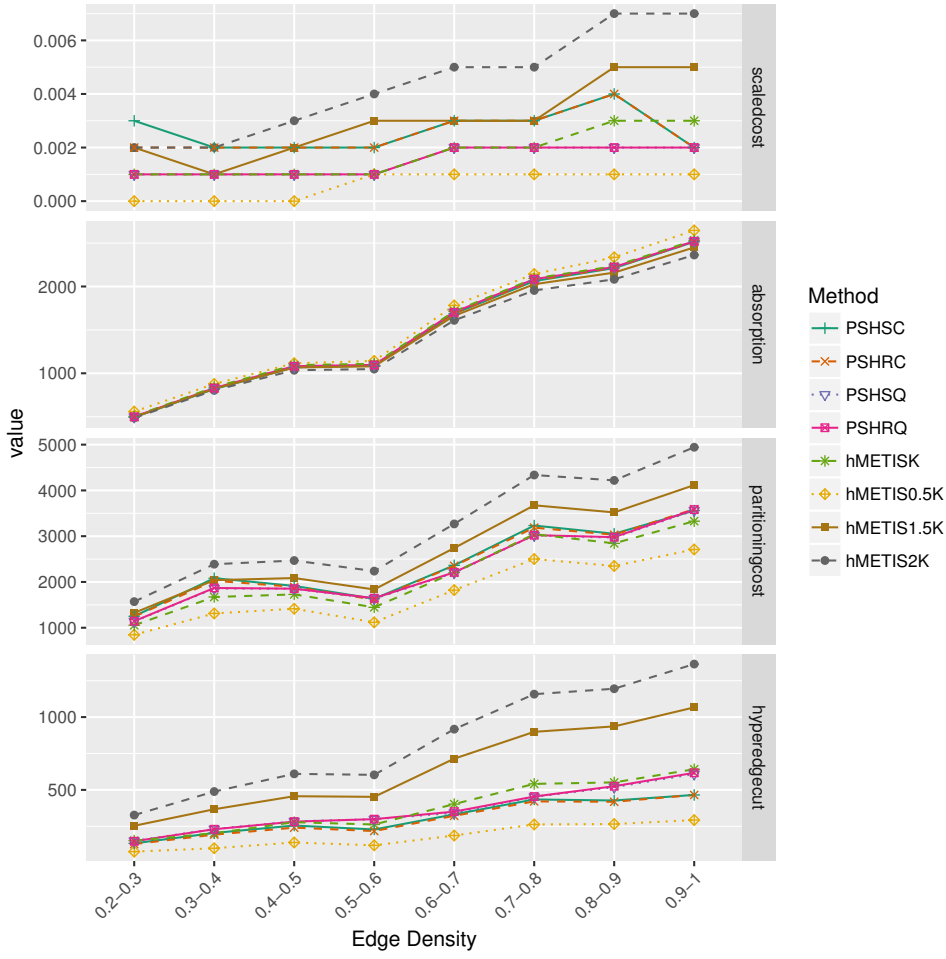


Figure 5.2: Global quality measures (scaled cost, absorption, partitioning cost, and hyperedge cut) for the various edge densities

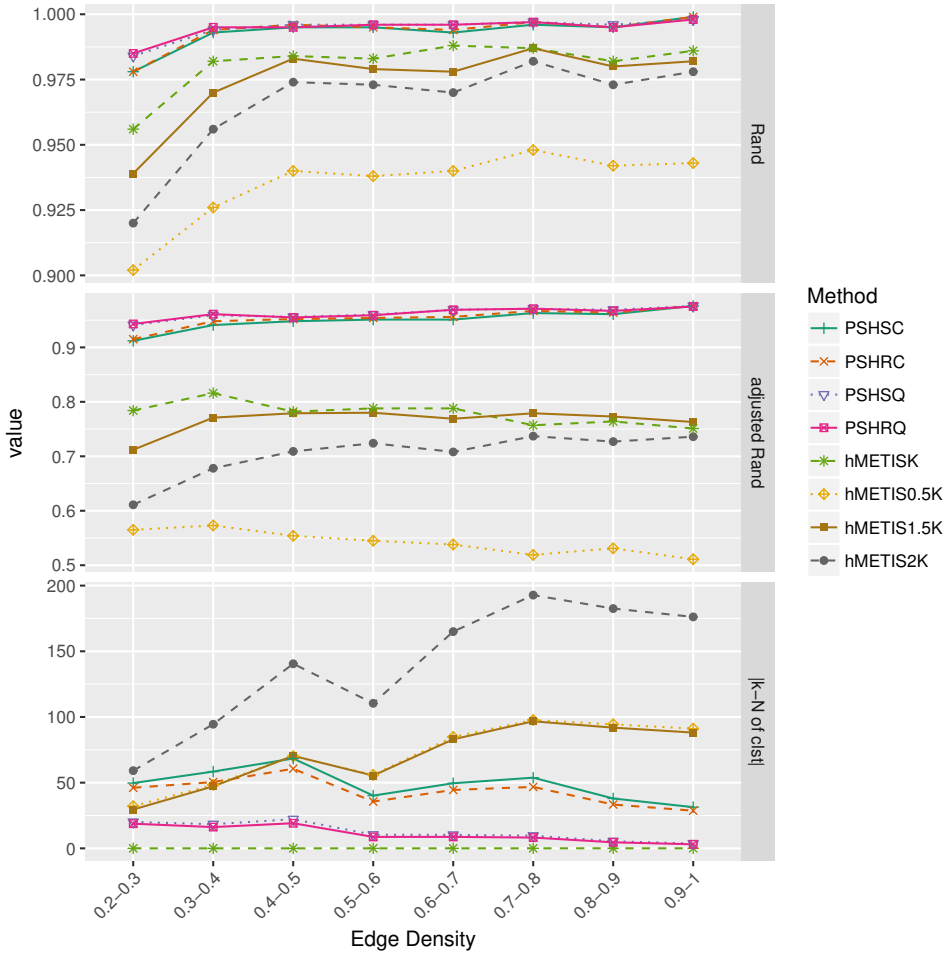


Figure 5.3: Validation quality measures (Rand index, adjusted Rand index, and $|k - \text{N.clst}|$) for the various edge densities

Table 5.3: Mean of running times of our algorithms (PSHSC, PSHRC, PSHSQ, and PSHRQ) for each n

	PSHSC		PSHRC		PSHSQ		PSHRQ	
n	Mean	SD	Mean	SD	Mean	SD	Mean	SD
100	0.85	0.46	0.72	0.42	0.27	0.12	0.20	0.09
200	1.12	0.58	1.00	0.49	0.49	0.20	0.44	0.21
500	1.92	0.86	1.89	0.86	1.53	0.80	1.57	0.90
1000	4.54	2.41	4.89	2.83	5.71	3.57	6.18	3.99
2000	12.28	7.38	13.98	8.74	19.01	10.89	20.79	12.27
5000	60.25	38.31	70.96	46.07	105.36	61.17	115.79	68.82
10000	268.04	179.35	319.87	217.79	486.32	294.33	538.34	332.14

the values $|k - \text{N.clst}|$ of the two algorithms PSHSQ and PSHRQ are quite a lot lower than those of the the other methods (PSHSC, PSHRC, hMETIS0.5K, hMETIS1.5K, and hMETIS2K) in all edge density ranges. In all edge density ranges, the Rand index and adjusted Rand index of our algorithms maintain high values. The differences in adjusted Rand index between our algorithms (PSHSC, PSHRC, PSHSQ, and PSHRQ) and hMETIS methods are larger than the differences in Rand index. The Rand index and adjusted Rand index of hMETIS0.5K are greatly far lower than those of our four algorithms (PSHSC, PSHRC, PSHSQ, and PSHRQ).

Figure 5.4 and Table 5.3 shows the mean and standard deviation (SD) of running times of our algorithms (PSHSC, PSHRC, PSHSQ, and PSHRQ) for each n. It is observed that the running times for $n > 2000$ rapidly increase. All algorithms contain a procedure to detect all maximal cliques in the line graph of a given hypergraph, and the complexity of the procedure exponentially increases as n increases. The two algorithms PSHSQ and PSHRQ need nearly twice as much running time as the other two algorithms PSHRC and PSHSC for large $n > 2000$, while the running times of PSHSQ and PSHRQ are shorter than those of PSHRC and PSHSC for small $n \geq 500$. Figure 5.5 shows that trend of running times as the edge density

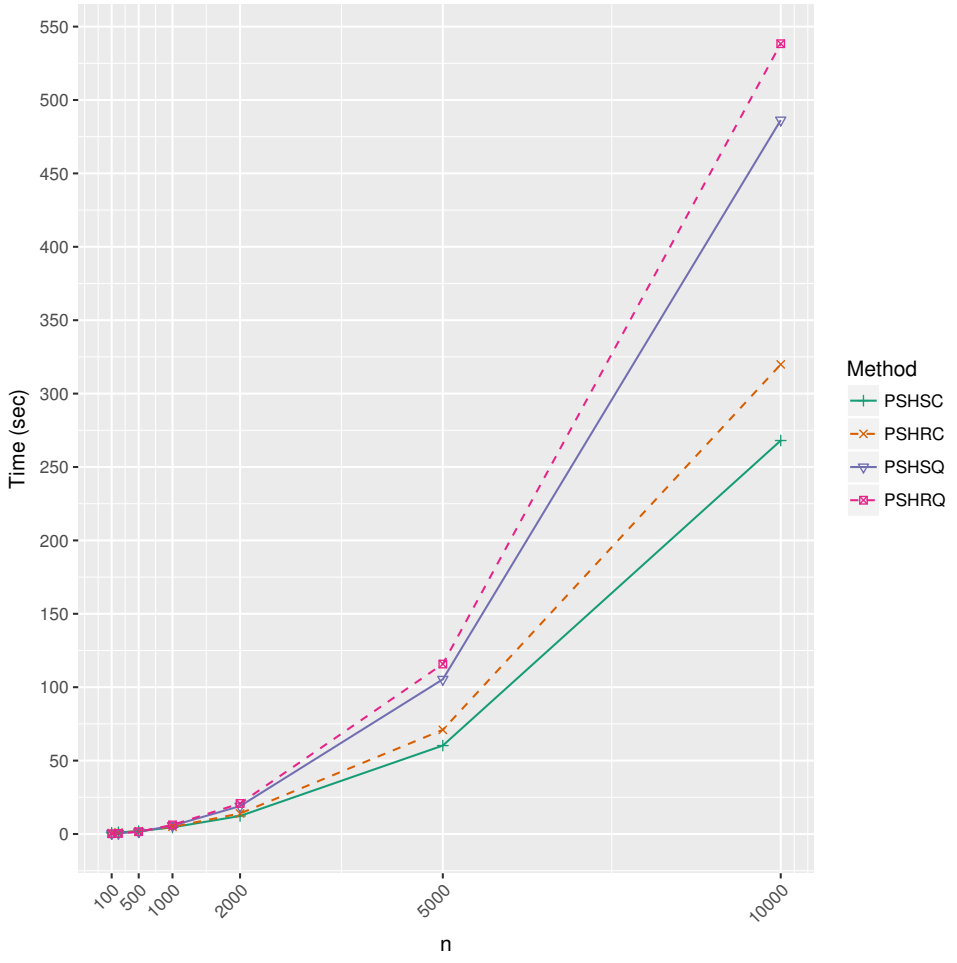


Figure 5.4: Mean of running times of our algorithms (PSHSC, PSHRC, PSHSQ, and PSHRQ) for each n

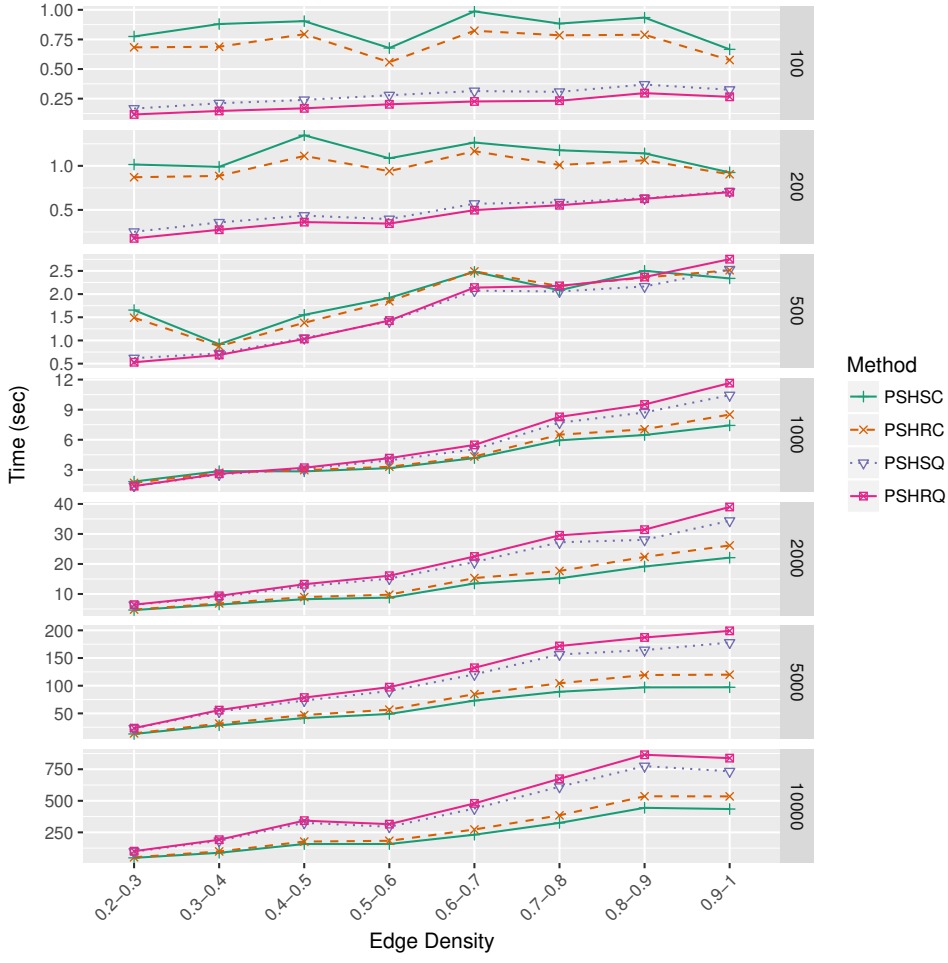


Figure 5.5: Trend of running times of our algorithms (PSHSC, PSHRC, PSHSQ, and PSHRQ) as the edge density increases

increases for each n. The running times tend to increase as the edge density increases even if the number of vertices are equal. We can observe that the algorithms PSHSQ and PSHRQ were affected by the edge density more than the other methods. For hypergraphs with $n = 10000$ and edge density from 0.8 to 1, the average running times of the algorithm PSHRQ are greater than 800 seconds, while it takes about 100 seconds for hypergraphs of edge density near 0.2. For all n, PSHRQ and PSHRC take longer running times than PSHSQ and PSHSC, relatively, since PSHRQ and PSHRC detect all cliques of the intersection graph. However, for the case $n = 10000$ and edge density is between 0.9 and 1, the difference of the mean of running time was only about 100 seconds. The hMETIS method we used (shmetis) is quite a fast program and every execution was completed in under 20 seconds. However, for the hypergraph with unknown cluster numbers and large n, it required repetitive executions to find an optimized solution.

5.5 Detecting protein complexes in PPI networks

5.5.1 Motivation

In this section, we attempt to predict protein complexes in a protein–protein interaction (PPI) network using the newly developed hypergraph clustering methods. Proteins are vital biomolecular components of organisms involved in vast and various functions occurring in cells. Thus, understanding protein functions is the key to understanding biological systems at the molecular level of life [NLC08]. Since proteins interact with each other in the regulation process for biological functions, rather than work as isolated entities, it has been a crucial issue in the study of proteomics to find protein complexes. PPIs can be represented using a graph (called a PPI network), where vertices represent proteins and two vertices are joined by

an edge if the corresponding two proteins interact. It has been revealed that functional complexes of proteins generally correspond to dense subgraphs of networks [TDN+02]. Thus, protein complexes can be detected by analyzing topological and structural properties of PPI networks. Moreover, owing to the development of sequencing technologies, large volumes of PPI networks have been integrated from the various datasets. There have been numerous attempts to detect protein complexes using computational methods to complement experimental methods which has several limitations in terms of time and cost, especially for large-scale PPI networks [JZL+14].

Many methods have been developed to predict protein complexes in a PPI network based on the graph-theoretical approaches [BH03, RG11, APF+06, AUASM+06, LFTN05, CNS+08, MTXY08, ADHJ09, ROP10, EZB12, LWC09, TK16]. There are roughly two types of methods that predict protein complexes in PPI networks. First, some of existing methods first detect locally dense subgraphs of a PPI network, and then merge the subgraphs according to the predefined rules [APF+06, ADHJ09, TK16, LFTN05, ROP10, LWC09]. The other methods search for a core vertex or initial clusters and then expand the core by adding highly connected nodes [BH03, RG11, AUASM+06, MTXY08]. We can consider the relatively denser subgraphs of the PPI networks, such as stars (set of a vertex and its neighbors) or cliques, as hyperedges and predict the protein complexes by merging them using the developed hypergraph clustering methods. Since one of the topological features of a PPI network is that a PPI network is a scale-free network in which the degree distribution follows a power law [JZL+14], we can effectively detect dense sub-networks and construct a hypergraph.

5.5.2 Methods

To apply the newly developed hypergraph clustering to PPI network analysis, we construct hyperedges from a PPI network in two ways: the first is to consider each vertex and its neighborhood (star) as a hyperedge and the second is to consider every maximal clique in a PPI network as a hyperedge. For the given PPI network, let H_{nbd} and H_{clique} be the hypergraphs constructed using the notion of neighborhood and clique, respectively. We applied PSHSC and PSHSQ algorithms to the PPI network data and compared the results. Since a protein can be involved in more than one functional module [JZL$^+$14], we developed a modified version of the PSHSQ algorithm (denoted by PSHSQ-OVL) that allows clusters to overlap. To allow clusters to overlap, we skip the refinement procedures of QSETCLUSTERING of the PSHSQ algorithm and take all clusters corresponding to the MWIS detected by the GWMIN algorithm as resulting clusters.

To compare with existing methods, we obtained the results of four methods MCODE [BH03], CFinder [APF$^+$06], PE-WCC [EZB12], and NCmine [TK16]. MCODE [BH03] detects locally dense subgraphs of a PPI network. The method first assigns weights to vertices based on the local neighborhood density and then, starting with the seed protein, expands the cluster to dense sub-networks by adding connected vertices with weights above a given threshold. CFinder [APF$^+$06] utilizes the notion of k-clique percolation and detects densely interconnected modules of PPI networks. PE-WCC [EZB12] first scores each edge in PPI networks by estimating the reliability of the PPI using the topological properties of the proteins joined by the edge and their neighborhood. After removing interactions with low weights, the algorithm detects complexes using a weighted clustering coefficient. NCmine [TK16] assigns a weight to each vertex based on degree centrality. Starting from the

vertex of highest weight, a local cluster of a vertex is selected to include the vertex and the subset of its neighbors with the edge density of the subgraph corresponding to this cluster exceeding a given threshold. Then overlapped clusters are merged if they satisfy the given conditions.

To estimate the degree of overlap between a predicted protein complex and a known protein complex, we used the Jaccard index defined as

$$Acc(C_p, C_k) = \frac{|C_p \cap C_k|}{|C_p \cup C_k|}$$

where C_p and C_k are the predicted and known protein complexes, respectively. If $Acc(C_p, C_k) > 0.5$, the predicted protein complex C_p is matched with the known protein complex C_k. To evaluate the performance of the prediction, we calculate the *recall (Rec)* and the *precision (Prec)* defined as follows:

$$Rec = \frac{N_{MK}}{N_K}$$

$$Prec = \frac{N_{MP}}{N_P}$$

Where N_{MK} and N_{MP} are the number of matching known complexes and the number of matching predicted complexes, respectively, and N_K and N_P are the number of known complexes and the number of predicted complexes, respectively.

For evaluation data, we used the *Saccharomyces cerevisiae* (yeast) PPI networks of the DIP dataset [XRS$^+$00] containing of 5017 proteins and 23115 interactions (released Oct. 1, 2014). For reference dataset of protein complexes, we used CYC2008 dataset [PWT$^+$08] that contains 408 protein complexes.

Table 5.4: Parameters that show the best performance of each method for predicting protein complexes in the DIP dataset regarding precision and recall.

Method	Parameter	Value
MCODE	degree cutoff	2
	node score cutoff	0.1
	K-core	3
CFinder	k	4
PE-WCC	join parameter	0.5
	overlap threshold	0.8
NCmine	cliqueness threshold	0.8
	merge threshold	0.8
	dCliqueness threshold	0.2
PSHSC	θ	0.8
PSHSQ	θ	0.9
PSHSQ-ovl	θ	0.9

5.5.3 Results

For MCODE, the maximum depth is set to 100, the degree of cutoff is set from 2 to 4, increased by 1, the node score cutoff is set to 0.1 or 0.2, and the K-core is set to 2 or 3. For CFinder, k is set to 3 or 4. For PE-WCC, the join parameter is set from 0.4 to 0.6, increased by 0.1, and the overlap threshold is set from 0.6 to 0.8, increased by 0.1. For NCmine, the cliqueness threshold is set from 0.4 to 0.8, increased by 0.2, the merge threshold is set from 0.4 to 0.8, increased by 0.2, the dCliqueness threshold is set from 0.2 or 0.4, and the cluster size threshold is set to 3.

For PSH methods (PSHSC, PSHSQ, and PSHSQ-OVL), the edge weight threshold (θ) for valid edges in a line graph of the constructed hypergraph is set from 0.6 to 0.9. Since the minimum size of the cluster detected by CFinder and PE-WCC is 3, for pair comparisons, we only consider the predicted and known complexes of a minimum size of 3. Among the protein complexes in the CYC2008 dataset, the number of known complexes of a minimum size of 3 is 149.

Figure 5.6 shows the recall and the precision calculated from the results obtained by existing methods and our PSH methods. MCODE, CFinder, PSHSC(H_{clique}),

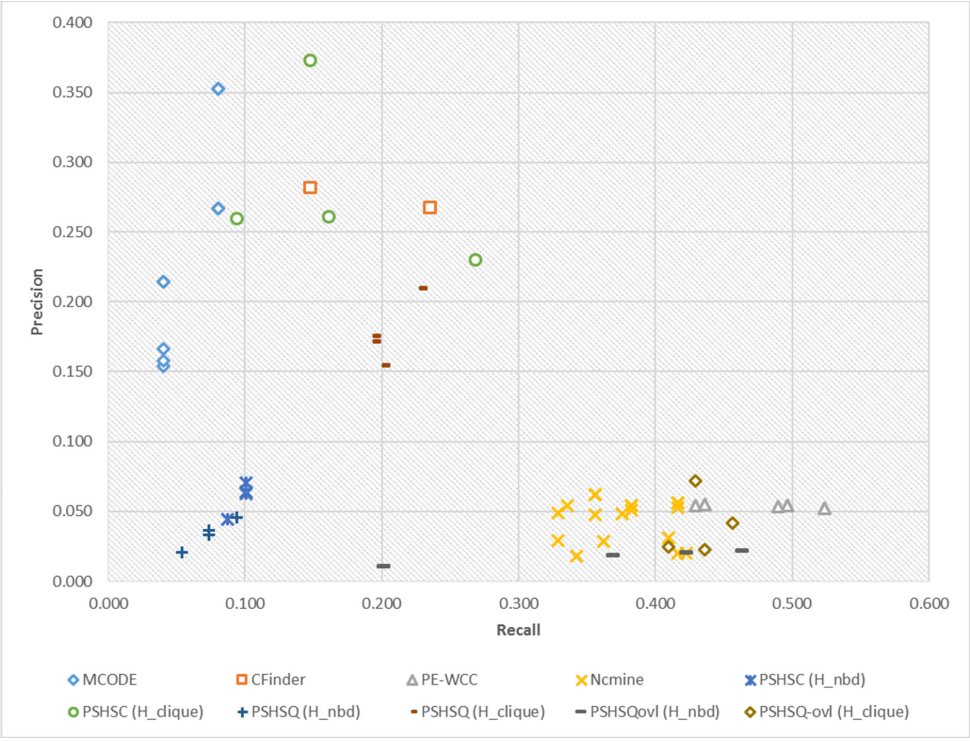


Figure 5.6: Recall and precision calculated from clustering results of existing methods (MCODE, CFinder, PE-WCC, and NCmine) and PSH methods (PSHSC, PSHSQ, and PSHSQ-OVL applied for H_{nbd} and H_{clique}).

Table 5.5: Comparisions of results obtained using the best parameters for each method.

Method	N. of matching complexes	N. of predicted complexes	Precision	Recall	Cluster size Mean (SD)
MCODE	12	34	0.353	0.081	12.62 (14.23)
CFinder	35	131	0.267	0.235	12.18 (28.08)
PE-WCC	78	1488	0.052	0.523	8.26 (6.26)
NCmine	62	1101	0.056	0.416	5.59 (2.07)
PSHSC (H_{nbd})	15	213	0.070	0.101	5.72 (2.84)
PSHSQ (H_{nbd})	14	305	0.046	0.094	12.49 (19.23)
PSHSQ-ovl (H_{nbd})	69	3107	0.022	0.463	14.59 (18.63)
PSHSC (H_{clique})	22	59	0.373	0.148	5.98 (2.48)
PSHSQ (H_{clique})	30	194	0.155	0.201	7.58 (6.67)
PSHSQ-ovl (H_{clique})	65	2825	0.023	0.436	4.73 (0.98)

and PSHSQ(H_{clique}) show better precision values than PE-WCC, NCmine, all PSH methods applied for H_{nbd}, and PSHSQ-OVL(H_{clique}). Meanwhile, PE-WCC, NCmine, and PSHSQ-OVL applied for H_{clique} and H_{nbd} show high recall values above 0.4 and, in particular, the recall values of PE-WCC are highest. Except for PE-WCC, the recall values of PSHSQ-OVL are largest; however, the precision values of this method are the smallest. PSHSC and PSHSQ applied for H_{clique} give relatively reasonable results, but PSH methods based on H_{nbd} show poor performance compared with the other methods.

For each method, we select one combination of parameters regarding the recall and the precision values (Table 5.4). The summary of best results of each method are shown in Table 5.5. The numbers of predicted complexes by MCODE, CFinder, and PSHSC(H_{clique}) are lower than the number of the known protein complexes, and the numbers of predicted complexes by PE-WCC, NCmine, PSHSQ-OVL(H_{nbd}), PSHSQ-OVL(H_{clique}) are much greater than the number of the known protein complexes. The method that detects the most matching complexes is PE-WCC, and PSHSQ-OVL(H_{nbd}) and PSHSQ-OVL(H_{clique}) follow. PSHSC and PSHSQ for H_{clique} predict fewer complexes than for H_{nbd}; nonetheless, the number of matching predicted complexes for H_{clique} by those methods are greater than for H_{nbd}. The mean and standard deviation of complex size are 8.83 and 9.26, respectively. The average cluster sizes of NCmine, PSHSC(H_{nbd}), PSHSC(H_{clique}), and PSHSQ-OVL(H_{clique}) are smaller than the average known complex size. Meanwhile, the average cluster sizes of MCODE, CFinder, PSHSQ(H_{nbd}), PSHSQ-OVL(H_{nbd}) are larger than the average known protein complex size.

5.5.4 Discussion

We applied several existing methods and our hypergraph clustering methods, PSH methods, to DIP yeast datasets and evaluated their performance using the reference dataset CYC2008 which contains the information of known protein complexes. The PSH methods showed much better performance for H_{clique} than for H_{nbd}. In particular, PSHSC(H_{clique}) showed the highest precision. PSHSQ-OVL detected the greatest number of known complexes, except for PE-WCC; however, they predicted too many complexes resulting in low precision. With this analysis, we could confirm that our algorithms can be used as a tool for detecting protein complexes with similar performance compared with other methods. Since our methods obtain the results only based on the topological structure of PPI networks, unlike the other methods that assign weights to vertices or edges to evaluate the centrality of proteins or reliability of PPIs, we expect to improve the performance of PSH methods on PPI networks by assigning weights to vertices or edges, and this remains for future study. By using a strategy such as rejecting unreliable clusters, the algorithms may be improved to detect overlapping protein complexes with higher precision.

5.6 Conclusions

The Big-LD algorithm is a method for partitioning the SNP sequence data into LD blocks. The algorithm captures the structure of the SNP sequence data using graph modeling and, based on that information, it finds clusters of unknown numbers and sizes. In the real word, there are many problems, such as the lack of pre-defined information in SNP sequence data, that make it more difficult to find the cluster solutions. For the data that have an underlying small community, information can be modeled using hypergraphs. However, most of the existing partitioning methods

of a hypergraph require pre-determined parameters such as the number of clusters. When we do not know the exact number of clusters *a priori*, we need additional treatment such as repeating the execution of the algorithm for a range of parameters to find the best solution.

In this context, we proposed new hypergraph partitioning algorithms (PSHSC, PSHRC, PSHSQ and PSHRQ) by generalizing the Big-LD algorithm. The new algorithms can be applied to sparse hypergraphs with unknown cluster structures. In the simulation study, we proved that the new algorithms are especially powerful in finding the hidden clustering structure of a sparse hypergraph. Although the algorithms tend to generate more clusters than the number of true underlying clusters, a high level of Rand index and adjusted Rand index show that the algorithms discover the true boundaries of the underlying clusters quite well.

In particular, the two algorithms PSHSQ and PSHRQ show the best-balanced performance regarding the local, global, and validation quality measures. The results of the two algorithms are better than the results of hMETIS (with $Npart$ of true cluster numbers) in terms of the local quality measures and the cluster validation measures. In addition, the number of partitions obtained by PSHSQ and PSHRQ was quite close to the number of true underlying clusters.

Two algorithms, PSHSC and PSHRC, showed good performance in detecting clusters with high connectivity. In other words, each vertex in a part constructed by PSHSC and PSHRC is well associated with edges in the part. In addition, they are faster than the other two algorithms, PSHSQ and PSHRQ.

Our algorithms consume more time for an execution compared with hMETIS; however, they require only one execution to find the underlying clustering structure while hMETIS requires additional treatment to find the true number of clusters if it is not known. In addition, for a very large number of vertices ($n = 10000$), the

execution of our algorithm is completed within a reasonable time (under 15 min).

Since new algorithms do not preset any parameters about the underlying true structure of the input data, our algorithms may not be very efficient at obtaining clusters of balanced sizes or a specific number. Our algorithms are designed for disjoint partitioning of the data, but we can obtain overlapping clusters by modifying the algorithm QSET-CLST by skipping the refinement procedures that remove overlaps.

For real-world data such as social network data (for example, co-authorship data or social tagging data) and biological network data (for example, SNP data, PPI network data, or gene–disease relation data), the algorithms suggested in this study may provide tools to reveal unknown structural characteristics. With the DIP yeast PPI network datasets, we confirmed that our algorithms can be used as a detection tool for protein complexes. In this application, we modified the PSHSQ algorithm to allow overlapping clusters (PSHSQ-OVL), and more matching complexes were detected by PSHSQ-OVL than the other PSH methods. We expect to improve the performance by adopting strategies to evaluate the centrality or reliability of the proteins or PPIs in a PPI network. In addition, we could develop more accurate versions of the algorithms that allow overlapping clusters by adding strategies to select reliable clusters.

Chapter 6

Conclusions

In this chapter, we summarize and discuss the findings of the thesis research. The research problem was clustering SNPs in SNP sequence data. The notion of cliques in the graph theory was used to determine SNP clusters in which every pair of SNPs is in strong LD. The CLQ algorithm assigned priority to the largest clique in the given data. The CLQ-D algorithm prioritized the maximal cliques in which the ratio of the number of SNPs to the physical range between the starting and ending SNPs. The CLQ algorithm has been implemented in multi-marker association methods [YSB13]. The CLQ-D algorithm modified from the CLQ algorithm has been used as a multi-level algorithm to identify hidden LD block structures of SNP sequence data. Cliques can be used for the first-level strategy to find SNP cluster, LD-bins, CLQ-D algorithm, but also for the second-level strategy to detect overlapping LD-bin structures of the interval graphs to identify LD blocks of SNP sequence data. With this multi-level strategy, LD blocks that allow "holes" [WP03] in the blocks can be identified. This approach is different from previously developed methods based on the statistical approach or haplotype pattern analysis. Moreover, the sparsity of data from the constructed graph model and the multi-level strategy facilitate the reduction of runtime and memory usage. This approach was implemented into the Big-LD algorithm. We showed that the LD block boundaries found by this algorithm are more invariant for the changes in the marker density compared to previous methods.

The underlying causes of LD block structure have been attributed to population

genetic phenomena, such as mutations, selection, recombination, or genetic drift. Presently, the LD block boundaries identified by the Big-LD algorithm displayed better agreement with the recombination hotspots compared to previous methods. It would be interesting to investigate the relationships between positive selection and big LD blocks identified by the Big-LD algorithm, since the long-range haplotypes are observed in big LD blocks more frequently, and because the long-range haplotypes provide evidence of positive selection. We analyzed big LD block patterns in the regions previously suggested [SVF$^+$07] to include candidate loci for positive selection. We provided evidence that the SLC30A9, PDE11A, and BCAS3 regions have been positively selected in both European and East Asians. Furthermore, the Big-LD algorithm was implicated as a useful detection tool of positive selection when comparing different populations. In future studies, the use of the Big-LD algorithm along with other statistically-based methods of positive selection detection could led to the development of a new method that will complement the previous methods.

The CLQ, CLQ-D, and Big-LD algorithms are novel means to partition SNP sequence data. The algorithms automatically determine the number of clusters and reveal the hidden structure. Analogous to the Big-LD algorithm, which is specialized for SNP data, we developed a new clustering (partitioning) algorithm based on a sparse hypergraph. We suggest that the four algorithms—PSHSC, PSHRC, PSHSQ, and PSHRQ—have different strategies (scope of candidate clusters and construction of intersection graphs) for sparse hypergraph clustering. In a simulation study, the algorithms partitioned sparse hypergraphs into high-quality clusters that were locally and globally balanced. The algorithms were effective at discerning the underlying true partitioning structure of hypergraphs. Our algorithms can be used to find structural characteristics of data with unordered data points, especially

for genetic data including protein-protein-interaction network data and gene-disease association data. We applied these algorithms to predict protein complexes in PPI network data set and confirmed their value as a clustering tool for protein complexes. In addition to the proposed algorithms in this research, we foresee some improvement in detecting overlapping clusters (such as protein complexes), which will be the subject of future reseach.

Chapter A

Coding correction algorithm

The sign of the correlation r_{ij} between two SNPs changes if we switch the risk allele and base allele for one of two SNPs. For example, if we replace $X_i$with a new genotype variable $X_i' = 2 - X_i$, then the genotype of X_i =0, 1, 2 becomes X_i' =2, 1, 0, respectively under an additive model. When this change is applied, the correlation between the genotype X_i' and X_j becomes $-r_{ij}$ for $i \neq j$. This coding change will also change the sign of beta estimates $\hat{\beta}_i$ if $\hat{\beta}_i \neq 0$. To make most pairwise correlations positive for SNPs in the joint analysis, we apply the Wang and Elston SNP recoding algorithm [WE07], which is as follows:

Step 1. Obtain the number of negatively correlated SNPs for each SNP i and denote it as n_i for $i = 1, 2, \cdots, m$, i.e. $n_i = \sum_{j=1, j\neq i}^{m} I(r_{ij} < 0)$ where I is an indicator function.

Step 2. Select the SNP with the $\max\{n_i\}$, then switch the risk and base allele for the genotype of that SNP.

Step 3. Iterate Steps 1-2 with updated correlations from the updated genotypes until $\max\{n_i\} < 2/m$.

References

[ADHJ09] Afnizanfaizal Abdullah, Safaai Deris, Siti Zaiton Mohd Hashim, and Hamimah Mohd Jamil. Graph partitioning method for functional module detections of protein interaction network. In *Computer Technology and Development, 2009. ICCTD'09. International Conference on*, volume 1, pages 230–234. IEEE, 2009.

[AER+04] Joshua M Akey, Michael A Eberle, Mark J Rieder, Christopher S Carlson, Mark D Shriver, Deborah A Nickerson, and Leonid Kruglyak. Population history and natural selection shape patterns of genetic variation in 132 genes. *PLoS Biol*, 2(10):e286, 2004.

[AHK98] Charles J Alpert, Jen-Hsin Huang, and Andrew B Kahng. Multilevel circuit partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(8):655–667, 1998.

[ALZM+05] Sameer Agarwal, Jongwoo Lim, Lihi Zelnik-Manor, Pietro Perona, David Kriegman, and Serge Belongie. Beyond pairwise clustering. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 838–845. IEEE, 2005.

[APF+06] Balázs Adamcsek, Gergely Palla, Illés J Farkas, Imre Derényi, and Tamás Vicsek. Cfinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, 2006.

[AST+12] Kristian G Andersen, Ilya Shylakhter, Shervin Tabrizi, Sharon R Grossman, Christian T Happi, and Pardis C Sabeti. Genome-wide scans provide evidence for positive selection of genes implicated in lassa fever. *Phil. Trans. R. Soc. B*, 367(1590):868–877, 2012.

[AUASM+06] Md Altaf-Ul-Amin, Yoko Shinbo, Kenji Mihara, Ken Kurokawa, and Shigehiko Kanaya. Development and implementation of an algorithm for detection of protein complexes in large interaction networks. *BMC bioinformatics*, 7(1):207, 2006.

[BBC02] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 238–247. IEEE, 2002.

[BCZ10] David H Ballard, Judy Cho, and Hongyu Zhao. Comparisons of multi-marker association methods to detect association between a candidate region and disease. *Genetic epidemiology*, 34(3):201–212, 2010.

[BFMD05] Jeffrey C Barrett, B Fry, JDMJ Maller, and Mark J Daly. Haploview: analysis and visualization of ld and haplotype maps. *Bioinformatics*, 21(2):263–265, 2005.

[BH03] Gary D Bader and Christopher WV Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC bioinformatics*, 4(1):2, 2003.

[BHG+16] Sean G Byars, Qinqin Huang, Lesley-Ann Gray, Samuli Ripatti, Gad Abraham, Stephen C Stearns, and Michael Inouye. Genetic loci associated with coronary artery disease harbor evidence of selection and antagonistic pleiotropy. *bioRxiv*, page 064758, 2016.

[BHP+08] Jarosław Bryk, Emilie Hardouin, Irina Pugach, David Hughes, Rainer Strotmann, Mark Stoneking, and Sean Myles. Positive selection in east asians for an edar allele that enhances nf- κ b activation. *PLoS One*, 3(5):e2209, 2008.

[BK73] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[BM12] William S Bush and Jason H Moore. Genome-wide association studies. *PLoS Comput Biol*, 8(12):e1002822, 2012.

[BMW+02] Maarit Bärlund, Outi Monni, J Donald Weaver, Päivikki Kauraniemi, Guido Sauter, Mervi Heiskanen, Olli-P Kallioniemi, and Anne Kallioniemi. Cloning of bcas3 (17q23) and bcas4 (20q13) genes that undergo amplification, overexpression, and fusion in

breast cancer. *Genes, Chromosomes and Cancer*, 35(4):311–317, 2002.

[BP09] Samuel R Bulò and Marcello Pelillo. A game-theoretic approach to hypergraph clustering. In *Advances in neural information processing systems*, pages 1571–1579, 2009.

[BTC+10] Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. Music recommendation by unified hypergraph: combining social media information and music content. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 391–400. ACM, 2010.

[C+10] International HapMap 3 Consortium et al. Integrating common and rare genetic variation in diverse human populations. *Nature*, 467(7311):52–58, 2010.

[C+12] 1000 Genomes Project Consortium et al. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, 2012.

[CCT+15] Christopher C Chang, Carson C Chow, Laurent CAM Tellier, Shashaank Vattikuti, Shaun M Purcell, and James J Lee. Second-generation plink: rising to the challenge of larger and richer datasets. *Gigascience*, 4(1):1, 2015.

[CDKU12] Ümit V Catalyürek, Mehmet Deveci, Kamer Kaya, and Bora Ucar. Multithreaded clustering for multi-level hypergraph partitioning. In *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 848–859. IEEE, 2012.

[CH69] Alan H Cheetham and Joseph E Hazel. Binary (presence-absence) similarity coefficients. *Journal of Paleontology*, pages 1130–1136, 1969.

[CLM99] A. Collins, C. Lonjou, and N. E. Morton. Genetic epidemiology of single-nucleotide polymorphisms. *Proceedings of the National Academy of Sciences*, 96(26):15173–15177, 1999.

[CN06] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*, 1695(5):1–9, 2006.

[CNS+08] Hon Nian Chua, Kang Ning, Wing-Kin Sung, Hon Wai Leong, and Limsoon Wong. Using indirect protein–protein interactions for protein complex prediction. *Journal of bioinformatics and computational biology*, 6(03):435–466, 2008.

[CPMC96] Tae-Soo Chon, Young Seuk Park, Kyong Hi Moon, and Eui Young Cha. Patternizing communities by using an artificial neural network. *Ecological modelling*, 90(1):69–78, 1996.

[CSZ94] P. K. Chan, M. D. F. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1088–1096, Sep 1994.

[dAH15] Renato Cordeiro de Amorim and Christian Hennig. Recovering the number of clusters in data sets with noise features using feature rescaling factors. *Information Sciences*, 324:126–145, 2015.

[DLS14] Andriy Derkach, Jerry F. Lawless, and Lei Sun. Pooled association tests for rare genetic variants: A review and some new results. pages 302–321, 2014.

[DPV05] Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique percolation in random networks. *Physical review letters*, 94(16):160202, 2005.

[DRS+01] Mark J. Daly, John D. Rioux, Stephen F. Schaffner, Thomas J. Hudson, and Eric S. Lander. High-resolution haplotype structure in the human genome. *Nat Genet*, 29(2):229–232, 2001.

[Edd13] Dirk Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer, New York, 2013. ISBN 978-1-4614-6867-7.

[EEM+15] Johannes Engelken, Guadalupe Espadas, Francesco M Mancuso, Nuria Bonet, Anna-Lena Scherr, Victoria Jímenez-Álvarez, Marta

Codina-Solà, Daniel Medina-Stacey, Nino Spataro, and Mark Stoneking. Signatures of evolutionary adaptation in quantitative trait loci influencing trace element homeostasis in liver. *Molecular biology and evolution*, page msv267, 2015.

[EF11] Dirk Eddelbuettel and Romain François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011.

[ELS10] David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In *International Symposium on Algorithms and Computation*, pages 403–414. Springer, 2010.

[ENK+07] Michael A Eberle, Pauline C Ng, Kenneth Kuhn, Lixin Zhou, Daniel A Peiffer, Luana Galver, Karine A Viaud-Martinez, Cynthia Taylor Lawley, Kevin L Gunderson, Richard Shen, et al. Power to detect risk alleles using genome-wide tag snp panels. *PLoS Genet*, 3(10):e170, 2007.

[EZB12] Dmitry Efimov, Nazar Zaki, and Jose Berengueres. Detecting protein complexes from noisy protein interaction data. In *Proceedings of the 11th International Workshop on Data Mining in Bioinformatics*, pages 1–7. ACM, 2012.

[FBC+07] Kelly A Frazer, Dennis G Ballinger, David R Cox, David A Hinds, Laura L Stuve, Richard A Gibbs, John W Belmont, Andrew Boudreau, Paul Hardenbol, Suzanne M Leal, et al. A second generation human haplotype map of over 3.1 million snps. *Nature*, 449(7164):851–861, 2007.

[Fea06] Paul Fearnhead. Sequenceldhot: detecting recombination hotspots. *Bioinformatics*, 22(24):3061–3066, 2006.

[FM83] Edward B Fowlkes and Colin L Mallows. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 78(383):553–569, 1983.

[FM88] Charles M Fiduccia and Robert M Mattheyses. A linear-time heuristic for improving network partitions. In *Papers on Twenty-five years of electronic design automation*, pages 241–247. ACM, 1988.

[FTT04] Gary William Flake, Robert E Tarjan, and Kostas Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408, 2004.

[GAS+13] Sharon R Grossman, Kristian G Andersen, Ilya Shlyakhter, Shervin Tabrizi, Sarah Winnicki, Angela Yen, Daniel J Park, Dustin Griesemer, Elinor K Karlsson, and Sunny H Wong. Identifying recent adaptations in large-scale genomic data. *Cell*, 152(4):703–713, 2013.

[GBH+03] Richard A Gibbs, John W Belmont, Paul Hardenbol, Thomas D Willis, Fuli Yu, Huanming Yang, Lan-Yang Ch'ang, Wei Huang, Bin Liu, and Yan Shen. The international hapmap project. *Nature*, 426(6968):789–796, 2003.

[GDG14] Suzanne Renick Gallagher, Micah Dombrower, and Debra S. Goldberg. Using 2-node hypergraph clustering coefficients to analyze disease-gene networks. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '14, pages 647–648, New York, NY, USA, 2014. ACM.

[GHHW00] Katrina AB Goddard, Penelope J Hopkins, Jeff M Hall, and John S Witte. Linkage disequilibrium and allele-frequency distributions for 114 single-nucleotide polymorphisms in five populations. *The American Journal of Human Genetics*, 66(1):216–234, 2000.

[GSK+10] Sharon R Grossman, Ilya Shylakhter, Elinor K Karlsson, Elizabeth H Byrne, Shannon Morales, Gabriel Frieden, Elizabeth Hostetter, Elaine Angelino, Manuel Garber, and Or Zuk. A composite of multiple signals distinguishes causal variants in regions of positive selection. *Science*, 327(5967):883–886, 2010.

[GSN+02] Stacey B Gabriel, Stephen F Schaffner, Huy Nguyen, Jamie M Moore, Jessica Roy, Brendan Blumenstiel, John Higgins, Matthew

DeFelice, Amy Lochner, and Maura Faggart. The structure of haplotype blocks in the human genome. *Science*, 296(5576):2225–2229, 2002.

[HA85] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[HKKM98] Eui-Hong Han, George Karypis, Vipin Kumar, and Bamshad Mobasher. Hypergraph based clustering in high-dimensional data sets: A summary of results. *IEEE Data Eng. Bull.*, 21(1):15–22, 1998.

[HKS05] Eran Halperin, Gad Kimmel, and Ron Shamir. Tag snp selection in genotype data for maximizing snp prediction accuracy. *Bioinformatics*, 21(suppl 1):i195–i203, 2005.

[HS00] Erez Hartuv and Ron Shamir. A clustering algorithm based on graph connectivity. *Information processing letters*, 76(4-6):175–181, 2000.

[HW79] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[HZPJ] Yi Han, Bin Zhou, Jian Pei, and Yan Jia. *Understanding Importance of Collaborations in Co-authorship Networks: A Supportiveness Analysis Approach*, pages 1112–1123.

[JKN01] Alec J Jeffreys, Liisa Kauppi, and Rita Neumann. Intensely punctate meiotic recombination in the class ii region of the major histocompatibility complex. *Nature genetics*, 29(2):217–222, 2001.

[JRN00] Alec J. Jeffreys, Alistair Ritchie, and Rita Neumann. High resolution analysis of haplotype diversity and meiotic crossover in the human tap2 recombination hotspot. *Human Molecular Genetics*, 9(5):725–733, 2000.

[JZL+14] Junzhong Ji, Aidong Zhang, Chunnian Liu, Xiaomei Quan, and Zhijun Liu. Survey: Functional module detection from protein-

protein interaction networks. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):261–277, 2014.

[KAKS99] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: applications in vlsi domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(1):69–79, 1999.

[Kar03] George Karypis. Multilevel hypergraph partitioning. In *Multilevel Optimization in VLSICAD*, pages 125–154. Springer, 2003.

[KHZ14] Soo-Jin Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Bayesian evolutionary hypergraph learning for predicting cancer clinical outcomes. *Journal of biomedical informatics*, 49:101–111, 2014.

[KK98a] George Karypis and Vipin Kumar. hmetis 1.5: A hypergraph partitioning package. Technical report, Technical report, Department of Computer Science, University of Minnesota, 1998. Available on the WWW at URL http://www. cs. umn. edu/metis, 1998.

[KK98b] George Karypis and Vipin Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 48(1):96–129, 1998.

[KK00] George Karypis and Vipin Kumar. Multilevel k-way hypergraph partitioning. *VLSI design*, 11(3):285–300, 2000.

[KLL+08] Lydia Coulter Kwee, Dawei Liu, Xihong Lin, Debashis Ghosh, and Michael P Epstein. A powerful and flexible multilocus association test for quantitative traits. *The American Journal of Human Genetics*, 82(2):386–397, 2008.

[KN+01] Leonid Kruglyak, Deborah A Nickerson, et al. Variation is the spice of life. *Nature genetics*, 27(3):234–235, 2001.

[KWR+01] Javed Khan, Jun S Wei, Markus Ringner, Lao H Saal, Marc Ladanyi, Frank Westermann, Frank Berthold, Manfred Schwab, Cristina R Antonescu, Carsten Peterson, et al. Classification and

diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature medicine*, 7(6):673–679, 2001.

[LEB+12] Seunggeun Lee, Mary J Emond, Michael J Bamshad, Kathleen C Barnes, Mark J Rieder, Deborah A Nickerson, ESP Lung Project Team, David C Christiani, Mark M Wurfel, Xihong Lin, et al. Optimal unified approach for rare-variant association testing with application to small-sample case-control whole-exome sequencing studies. *The American Journal of Human Genetics*, 91(2):224–237, 2012.

[Lew64] R. C. Lewontin. The interaction of selection and linkage. i. general considerations; heterotic models. *Genetics*, 49(1):49–67, 1964.

[Lew95] RC Lewontin. The detection of linkage disequilibrium in molecular sequence data. *Genetics*, 140(1):377–388, 1995.

[LFTN05] Xiao-Li Li, Chuan-Sheng Foo, Soon-Heng Tan, and See-Kiong Ng. Interaction graph mining for protein complexes using local clique merging. *Genome Informatics*, 16(2):260–269, 2005.

[LH02] Martin J Lercher and Laurence D Hurst. Human snp variability and mutation rate are higher in regions of high recombination. *Trends in genetics*, 18(7):337–340, 2002.

[LJ15] Foad Lotfifar and Matthew Johnson. A multi–level hypergraph partitioning algorithm using rough set clustering. In *European Conference on Parallel Processing*, pages 159–170. Springer, 2015.

[LLM10] Jure Leskovec, Kevin J Lang, and Michael Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM, 2010.

[LLY15] Hairong Liu, Longin Jan Latecki, and Shuicheng Yan. Dense subgraph partition of positive hypergraphs. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):541–554, 2015.

[LWC09] Guimei Liu, Limsoon Wong, and Hon Nian Chua. Complex discovery from weighted ppi networks. *Bioinformatics*, 25(15):1891–1897, 2009.

[Man10] Teri A. Manolio. Genomewide association studies and assessment of the risk of disease. *New England Journal of Medicine*, 363(2):166–176, 2010.

[McD09] John H McDonald. *Handbook of biological statistics*, volume 2. Sparky House Publishing Baltimore, MD, 2009.

[MMH+04] Gilean A. T. McVean, Simon R. Myers, Sarah Hunt, Panos Deloukas, David R. Bentley, and Peter Donnelly. The fine-scale structure of recombination rate variation in the human genome. *Science*, 304(5670):581–584, 2004.

[Mor05] Newton E Morton. Linkage disequilibrium maps and association mapping. *The Journal of clinical investigation*, 115(6):1425–1430, 2005.

[MTXY08] Mutlu Mete, Fusheng Tang, Xiaowei Xu, and Nurcan Yuruk. A structural approach for finding functional modules from large biological networks. *Bmc Bioinformatics*, 9(9):S19, 2008.

[New04] Mark EJ Newman. Coauthorship networks and patterns of scientific collaboration. *Proceedings of the national academy of sciences*, 101(suppl 1):5200–5205, 2004.

[NLC08] David L Nelson, Albert L Lehninger, and Michael M Cox. *Lehninger principles of biochemistry*. Macmillan, 2008.

[NRV+11] Benjamin M. Neale, Manuel A. Rivas, Benjamin F. Voight, David Altshuler, Bernie Devlin, Marju Orho-Melander, Sekar Kathiresan, Shaun M. Purcell, Kathryn Roeder, and Mark J. Daly. Testing for an unusual distribution of rare variants. *PLoS Genet*, 7(3):e1001322, 2011.

[NS04] Benjamin M. Neale and Pak C. Sham. The future of association studies: Gene-based analysis and replication. *The American Journal of Human Genetics*, 75(3):353–362, 2004.

[OB12] Peter Ochs and Thomas Brox. Higher order motion models and spectral clustering. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 614–621. IEEE, 2012.

[Och57] Akira Ochiai. Zoogeographic studies on the soleoid fishes found in japan and its neighbouring regions. *Bull. Jpn. Soc. Sci. Fish*, 22(9):526–530, 1957.

[Ott00] Jurg Ott. Predicting the range of linkage disequilibrium. *Proceedings of the National Academy of Sciences*, 97(1):2–3, 2000.

[Pan09] Wei Pan. Asymptotic tests of association with multiple snps in linkage disequilibrium. *Genetic Epidemiology*, 33(6):497–507, 2009.

[PBH+01] Nila Patil, Anthony J Berno, David A Hinds, Wade A Barrett, Jigna M Doshi, Coleen R Hacker, Curtis R Kautzer, Danny H Lee, Claire Marjoribanks, David P McDonough, et al. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, 294(5547):1719–1723, 2001.

[PDFV05] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[PDL+13] Marc Pybus, Giovanni M Dall'Olio, Pierre Luisi, Manu Uzkudun, Angel Carreno-Torres, Pavlos Pavlidis, Hafid Laayouni, Jaume Bertranpetit, and Johannes Engelken. 1000 genomes selection browser 1.0: a genome browser dedicated to signatures of natural selection in modern humans. *Nucleic acids research*, page gkt1188, 2013.

[Pet68] James A Peters. A computer program for calculating degree of biogeographical resemblance between areas. *Systematic Biology*, 17(1):64–69, 1968.

[PP01] Jonathan K Pritchard and Molly Przeworski. Linkage disequilibrium in humans: models and data. *The American Journal of Human Genetics*, 69(1):1–14, 2001.

[PRFP08] Cristian Pattaro, Ingo Ruczinski, Daniele M Fallin, and Giovanni Parmigiani. Haplotype block partitioning as a tool for dimensionality reduction in snp association studies. *BMC genomics*, 9(1):405, 2008.

[PWT+08] Shuye Pu, Jessica Wong, Brian Turner, Emerson Cho, and Shoshana J Wodak. Up-to-date catalogues of yeast protein complexes. *Nucleic acids research*, 37(3):825–831, 2008.

[R C17] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017.

[Ran71] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[RB03] J Kay Richmond and Deborah J Baglole. Lassa fever: epidemiology, clinical features, and social consequences. *BMJ: British Medical Journal*, 327(7426):1271, 2003.

[RCB+01a] David E. Reich, Michele Cargill, Stacey Bolk, James Ireland, and et al. Linkage disequilibrium in the human genome. *Nature*, 411(6834):199–204, May 10 2001. Copyright - Copyright Macmillan Journals Ltd. May 10, 2001; Last updated - 2013-01-27; CODEN - NATUAS.

[RCB+01b] David E. Reich, Michele Cargill, Stacey Bolk, James Ireland, Pardis C. Sabeti, Daniel J. Richter, Thomas Lavery, Rose Kouyoumjian, Shelli F. Farhadian, Ryk Ward, and Eric S. Lander. Linkage disequilibrium in the human genome. *Nature*, 411(6834):199–204, 2001.

[REW+04] Naheed A. Rana, Neil D. Ebenezer, Andrew R. Webster, Andres R. Linares, David B. Whitehouse, Sue Povey, and Alison J. Hardcastle. Recombination hotspots and block structure of linkage disequilibrium in the human genome exemplified by detailed analysis of pgm1 on 1p31. *Human Molecular Genetics*, 13(24):3089–3102, 2004.

[RG11] Kahn Rhrissorrakrai and Kristin C Gunsalus. Mine: module identification in networks. *BMC bioinformatics*, 12(1):192, 2011.

[ROP10] Emad Ramadan, Christopher Osgood, and Alex Pothen. Discovering overlapping modules and bridge proteins in proteomic networks. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, pages 366–369. ACM, 2010.

[RPT10] Emad Ramadan, Sudhir Perincheri, and David Tuck. A hyper-graph approach for analyzing transcriptional networks in breast cancer. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, pages 556–562. ACM, 2010.

[Sch07] Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

[Sla08] Montgomery Slatkin. Linkage disequilibrium—understanding the evolutionary past and mapping the medical future. *Nature Reviews Genetics*, 9(6):477–485, 2008.

[SM00] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug 2000.

[SMB+07] Laura J Scott, Karen L Mohlke, Lori L Bonnycastle, Cristen J Willer, Yun Li, William L Duren, Michael R Erdos, Heather M Stringham, Peter S Chines, Anne U Jackson, et al. A genome-wide association study of type 2 diabetes in finns detects multiple susceptibility variants. *science*, 316(5829):1341–1345, 2007.

[SRH+02] Pardis C Sabeti, David E Reich, John M Higgins, Haninah ZP Levine, Daniel J Richter, Stephen F Schaffner, Stacey B Gabriel, Jill V

Platko, Nick J Patterson, and Gavin J McDonald. Detecting recent positive selection in the human genome from haplotype structure. *Nature*, 419(6909):832–837, 2002.

[SS+63] Robert R Sokal, Peter HA Sneath, et al. Principles of numerical taxonomy. *Principles of numerical taxonomy.*, 1963.

[SS95] Wern-Jieh Sun and C. Sechen. Efficient and effective placement for very large circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(3):349–359, Mar 1995.

[SS05] Matthew Stephens and Paul Scheet. Accounting for decay of linkage disequilibrium in haplotype inference and missing-data imputation. *The American Journal of Human Genetics*, 76(3):449–462, 2005.

[SSD01] Matthew Stephens, Nicholas J Smith, and Peter Donnelly. A new statistical method for haplotype reconstruction from population data. *The American Journal of Human Genetics*, 68(4):978–989, 2001.

[SSF+06] Pardis C Sabeti, Stephen F Schaffner, Ben Fry, Jason Lohmueller, Patrick Varilly, Oleg Shamovsky, Alejandro Palma, TS Mikkelsen, D Altshuler, and ES Lander. Positive natural selection in the human lineage. *science*, 312(5780):1614–1620, 2006.

[Str04] Daniel O Stram. Tag snp selection for association studies. *Genetic epidemiology*, 27(4):365–374, 2004.

[Str08] Constantine A Stratakis. *Cushing syndrome caused by adrenocortical tumors and hyperplasias (corticotropin-independent Cushing syndrome)*, volume 13, pages 117–132. Karger Publishers, 2008.

[STY03] Shuichi Sakai, Mitsunori Togasaki, and Koichi Yamazaki. A note on greedy algorithms for the maximum weighted independent set problem. *Discrete Applied Mathematics*, 126(2):313–322, 2003.

[SVF+07] Pardis C. Sabeti, Patrick Varilly, Ben Fry, Jason Lohmueller, Elizabeth Hostetter, Chris Cotsapas, Xiaohui Xie, Elizabeth H. Byrne,

Steven A. McCarroll, Rachelle Gaudet, Stephen F. Schaffner, and Eric S. Lander. Genome-wide detection and characterization of positive selection in human populations. *Nature*, 449(7164):913–918, 2007.

[SXZF08] Jimeng Sun, Yinglian Xie, Hui Zhang, and Christos Faloutsos. Less is more: Sparse graph mining with compact matrix decomposition. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 1(1):6–22, 2008.

[Tak82] Naoyuki Takahata. Linkage disequilibrium, genetic distance and evolutionary distance under a general model of linked genes or a part of the genome. *Genetics Research*, 39(01):63–77, 1982.

[TBCH11] Shulong Tan, Jiajun Bu, Chun Chen, and Xiaofei He. Using rich social media information for music recommendation via hypergraph model. In *Social media modeling and computing*, pages 213–237. Springer, 2011.

[TDN+02] Amy Hin Yan Tong, Becky Drees, Giuliano Nardelli, Gary D Bader, Barbara Brannetti, Luisa Castagnoli, Marie Evangelista, Silvia Ferracuti, Bryce Nelson, Serena Paoluzi, et al. A combined experimental and computational strategy to define protein interaction networks for peptide recognition modules. *Science*, 295(5553):321–324, 2002.

[TGLP16] Daniel Taliun, Johann Gamper, Ulf Leser, and Cristian Pattaro. Fast sampling-based whole-genome haplotype block recognition. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(2):315–325, 2016.

[TGP14] Daniel Taliun, Johann Gamper, and Cristian Pattaro. Efficient haplotype block recognition of very long and dense genetic sequences. *BMC bioinformatics*, 15(1):1, 2014.

[THK09] Ze Tian, TaeHyun Hwang, and Rui Kuang. A hypergraph-based learning algorithm for classifying gene expression and arraycgh

data with prior knowledge. *Bioinformatics*, 25(21):2831–2838, 2009.

[TIAS77] Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6(3):505–517, 1977.

[TK16] Shu Tadaka and Kengo Kinoshita. Ncmine: Core-peripheral based functional module detection using near-clique mining. *Bioinformatics*, 32(22):3454–3460, 2016.

[TMP+03] Rebecca CJ Twells, Charles A Mein, Michael S Phillips, J Fred Hess, Riitta Veijola, Matthew Gilbey, Matthew Bright, Michael Metzker, Benedicte A Lie, and Amanda Kingsnorth. Haplotype structure, ld blocks, and uneven recombination within the lrp5 gene. *Genome research*, 13(5):845–855, 2003.

[VKWP06] Benjamin F Voight, Sridhar Kudaravalli, Xiaoquan Wen, and Jonathan K Pritchard. A map of recent positive selection in the human genome. *PLoS Biol*, 4(3):e72, 2006.

[WAZ+02] Ning Wang, Joshua M Akey, Kun Zhang, Ranajit Chakraborty, and Li Jin. Distribution of recombination crossovers and the origin of haplotype blocks: the interplay of population history, recombination, and mutation. *The American Journal of Human Genetics*, 71(5):1227–1234, 2002.

[WC84] Bruce S Weir and C Clark Cockerham. Estimating f-statistics for the analysis of population structure. *evolution*, pages 1358–1370, 1984.

[WE07] Tao Wang and Robert C Elston. Improved power by use of a weighted score test for linkage disequilibrium mapping. *The american journal of human genetics*, 80(2):353–360, 2007.

[WEG87] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[WHC+07] Scott H Williamson, Melissa J Hubisz, Andrew G Clark, Bret A Payseur, Carlos D Bustamante, and Rasmus Nielsen. Localizing recent adaptive evolution in the human genome. *PLoS Genet*, 3(6):e90, 2007.

[WKE+10] Michael C. Wu, Peter Kraft, Michael P. Epstein, Deanne M. Taylor, Stephen J. Chanock, David J. Hunter, and Xihong Lin. Powerful snp-set analysis for case-control genome-wide association studies. *The American Journal of Human Genetics*, 86(6):929–942, 2010.

[WLC+11] Michael C Wu, Seunggeun Lee, Tianxi Cai, Yun Li, Michael Boehnke, and Xihong Lin. Rare-variant association testing for sequencing data with the sequence kernel association test. *The American Journal of Human Genetics*, 89(1):82–93, 2011.

[WLZ15] Qian Wang, Qiongshi Lu, and Hongyu Zhao. A review of study designs and statistical methods for genomic epidemiology studies using next generation sequencing. *Frontiers in Genetics*, 6, 2015.

[WP03] Jeffrey D Wall and Jonathan K Pritchard. Assessing the performance of the haplotype block model of linkage disequilibrium. *The American Journal of Human Genetics*, 73(3):502–515, 2003.

[XRS+00] Ioannis Xenarios, Danny W Rice, Lukasz Salwinski, Marisa K Baron, Edward M Marcotte, and David Eisenberg. Dip: the database of interacting proteins. *Nucleic acids research*, 28(1):289–291, 2000.

[XSD+02] Ioannis Xenarios, Lukasz Salwinski, Xiaoqun Joyce Duan, Patrick Higney, Sul-Min Kim, and David Eisenberg. Dip, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic acids research*, 30(1):303–305, 2002.

[YKB15] Yun Joo Yoo, Sun Ah Kim, and Shelley B. Bull. Clique-based clustering of correlated snps in a gene can improve performance of gene-based multi-bin linear combination test. *BioMed Research International*, 2015:11, 2015.

[YSB13] Yun Joo Yoo, Lei Sun, and Shelley B. Bull. Gene-based multiple regression association testing for combined examination of common and low frequency variants in quantitative trait analysis. *Frontiers in Genetics*, 4, 2013.

[YSP+17] Yun Joo Yoo, Lei Sun, Julia G Poirier, Andrew D Paterson, and Shelley B Bull. Multiple linear combination (mlc) regression tests for common variants adapted to linkage disequilibrium structure. *Genetic epidemiology*, 41(2):108–121, 2017.

[Zap11] Carlos Zapata. On the uses and applications of the most commonly used measures of linkage disequilibrium from the comparative analysis of their statistical properties. *Human heredity*, 71(3):186–195, 2011.

[ZCY09] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, 2(1):718–729, 2009.

[ZDC+02] Kui Zhang, Minghua Deng, Ting Chen, Michael S Waterman, and Fengzhu Sun. A dynamic programming algorithm for haplotype block partitioning. *Proceedings of the National Academy of Sciences*, 99(11):7335–7339, 2002.

[ZGS+11] Jingyuan Zhao, Simone Gupta, Mark Seielstad, Jianjun Liu, and Anbupalam Thalamuthu. Pathway-based analysis using reduced gene subsets in genome-wide association studies. *BMC Bioinformatics*, 12(1):1–14, 2011.

[ZHS06] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems (NIPS) 19*, page 2006. MIT Press, 2006.

[ZL10] Zi-Ke Zhang and Chuang Liu. A hypergraph model of social tagging networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(10):P10005, 2010.

[ZLT+15] Chao Zhang, Jing Li, Lei Tian, Dongsheng Lu, Kai Yuan, Yuan Yuan, and Shuhua Xu. Differential natural selection of human zinc transporter genes between african and non-african populations. *Scientific reports*, 5:9658, 2015.

[ZQL+04] Kui Zhang, Zhaohui S Qin, Jun S Liu, Ting Chen, Michael S Waterman, and Fengzhu Sun. Haplotype block partitioning and tag snp selection using genotype data and their applications to association studies. *Genome Research*, 14(5):908–916, 2004.

[ZRM+05] Eleftheria Zeggini, William Rayner, Andrew P Morris, Andrew T Hattersley, Mark Walker, Graham A Hitman, Panos Deloukas, Lon R Cardon, and Mark I McCarthy. An evaluation of hapmap sample size and tagging snp performance in large-scale empirical and simulated data sets. *Nature genetics*, 37(12):1320–1322, 2005.

국문초록

클러스터링은 데이터의 의미있는 패턴을 밝혀내는 방법으로써 가장 널리 쓰이는 수단 중 하나이다. 최근 염기서열 데이터의 양이 방대하게 늘어나면서, 각 데이터의 구조에 적합한 효과적인 클러스터링 툴을 개발하는 것이 유전체학의 주요한 과제로 대두되었다. 본고에서는 그래프 이론을 기반으로 하여 SNP 데이터를 분할하는 알고리즘 CLQ 와 CLQ-D를 개발하였다. 또한, 이 알고리즘들을 이용하여 SNP 데이터의 클러스터링 결과를 얻은 후, 이 결과를 인터벌 그래프로 모델링하여 SNP 서열 데이터의 LD 블록을 탐지하는 알고리즘 Big-LD 를 개발하였다. 희박한 그래프란 그래프 내의 변의 수가 생성가능한 변의 수보다 훨씬 적은 그래프이다. 생물학적, 사회학적, 그리고 인터넷 네트워크 데이터와 같은 실세계 데이터가 희박한 그래프로 모델링 될 수 있다. SNP 클러스터링 알고리즘 CLQ, CLQ-D와 LD 블록 탐지 알고리즘 Big-LD는 주어진 SNP 데이터를 그래프로 모델링 하는데, 이 때 모델링 된 그래프는 SNP 데이터의 특수한 구조적 성격으로 인해 희박한 그래프의 구조를 갖게 된다. 이는 실행 시간 및 메모리 사용 측면에서 알고리즘이 효율적으로 동작할 수 있도록 한다. 특히, Big-LD 알고리즘은 “hole”을 포함하는 특별한 형태의 LD 블록을 탐지할 수 있으며, 이는 이전에 개발되었던 방법들이 탐지되지 않던 구조이다. 본고에서는 Big-LD 알고리즘을 HapMap phase 3 데이터와 1000 Genomes Project의 phase 1 데이터에 적용하여 각 데이터의 LD 블록 구조를 탐지하고, 이 결과를 이용하여 규모가 큰 LD 블록 구조와 생물학적 현상간의 관련성을 연구하였다. Big-LD 알고리즘으로 탐지된 LD 블록의 경계는 다른 방법으로 탐지된 LD 블록 경계과 비교해 보았을 때, 더 높은 비율로 유전자의 재조합이 빈번하게 일어나는 핫스폿의 위치와 일치한다. 또한 양성 선택 현상과 관련하여 인종간의 LD 블록 구조의 비교 분석을 시행하였으며, 이를 통해 선행 연구에서 양성 선택의 후보 위치로 제시되었던 유전자 지역들에 대하여 추가적인 근거를 제시하였다. 본고에서는 SNP 데이터를 LD

블록으로 나누는 Big-LD 알고리즘을 일반화함으로써 희박한 하이퍼그래프를 클러스터링하는 네 가지의 알고리즘—PSHSC, PSHRC, PSHSQ, PSHRQ—을 제안하였다. 모의실험을 통하여, 제안한 알고리즘이 모의생성된 하이퍼그래프의 실제 클러스터 구조에 매우 근접할 뿐만 아니라, 전역적, 지역적 성능 측정 수치로 볼 때 우수한 클러스터링 결과를 생성함을 보였다. 또한 실세계 데이터인 효모 단백질-단백질 상호작용 네트워크 데이터에 하이퍼그래프 클러스터링 알고리즘을 적용하여 단백질 복합체를 예측하는 실험을 통하여 생물학적 네트워크 데이터를 클러스터링하는 툴로서의 잠재성을 확인하였다.

주요어 : 하이퍼 그래프, 클러스터링, 최대가중치독립집합, 완전부분그래프, 연관비평형,연관비평형블록, 단일염기 다형성, 양성 선택

학번 : 2009-21501