unibz
Fakultät für Informatik
Facoltà di Scienze e Tecnologie informatiche
Faculty of Computer Science

Efficient Whole Genome Haplotype Block Partitioning using Linkage Disequilibrium

Author:

Daniel Taliun

Supervisors:

Prof. Johann Gamper
Ph.D. Cristian Pattaro

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Faculty of Computer Science at Free University of Bozen-Bolzano, Italy

March 31, 2015

Abstract

Investigation of linkage-disequilibrium (LD) patterns and underlying haplotype blocks across large genomic segments is a key element of genetic investigations over the whole genome. LD-based haplotype block partitions can be used in genome-wide haplotype association studies, set-based analyses where adjacent single nucleotide polymorphisms (SNPs) are collapsed together, cross-ethnicity comparisons, downstream analyses and interpretation of genome-wide association studies (GWASs). However, the LD-based haplotype block partitioning in large and dense genomic segments has always been a challenge due to the poor computational scalability of available algorithms. Moreover, the new sequencing technologies allow generating datasets with millions of SNPs, which are an order of magnitude larger than previously available.

In this work, we introduce a series of new optimizations of the most widely used LD-based haplotype block partitioning algorithm which was proposed by Gabriel *et al.* in 2002. We developed MIG^{++} – a memory and time efficient implementation of the Gabriel *et al.* algorithm. MIG^{++} has only $\Theta(n)$ memory complexity and by $>$80% improved runtime compared to the original algorithm, which has $\Theta(n^2)$ memory and runtime complexities. MIG^{++} incrementally constructs haplotype blocks and applies sophisticated search space pruning. We theoretically proved that the applied pruning preserves all blocks and experimentally showed that it omits $>$80% of computations. Differently from the existing software, MIG^{++} avoids restrictions on the maximal block length, considers SNP pairs at any distance, and can handle any number of SNPs.

The MIG^{++} runtime was further improved by replacing the standard likelihood-based LD variance estimator with an approximated estimator. With the likelihood-based method, the haplotype block partition of the entire HapMap II CEPH dataset was obtained in 457 hours. Meanwhile, the approximate method obtained the full-genome haplotype block partition of the entire 1000 Genomes Project CEPH dataset only in 44 hours, with no restrictions on allele frequency or long-range correlations. However, compared to the standard likelihood-based approach, with the approximate method rare adjacent SNPs tend to be clustered together into wider haplotype blocks leading to a coarser partition. These genome-wide experiments

also showed that LD-based haplotype blocks can span more than one million base-pairs in both HapMapII and 1000 Genomes Project datasets.

In order to avoid approximated LD variance estimator, we proposed a new sampling-based algorithm, called S-MIG^{++}, where the main idea is to estimate the area that is most likely to contain all the haplotype blocks by sampling a very small number of SNP pairs. A subsequent refinement step computes the exact blocks by considering only the SNP pairs within the estimated area. This approach significantly reduces the number of computations making the recognition of haplotype blocks very fast. We theoretically and empirically prove that the area containing all the haplotype blocks can be estimated with very high degree of certainty. Through experiments on 243,080 SNPs on chromosome 20 from the 1000 Genomes Project, we compared MIG^{++} with S-MIG^{++} and observed runtime reduction from 2.8 weeks to 34.8 hours. In a parallelized version of the S-MIG^{++} algorithm using 32 parallel processes, the runtime was further reduced to 5.1 hours.

The proposed MIG^{++} and S-MIG^{++} algorithms enable to perform LD-based haplotype block partitioning on genetic sequences of any length and density, which was previously infeasible. In the new generation sequencing era, this can help identify haplotypes that carry rare variants of interest. The low computational requirements open the possibility to include the haplotype block structure into GWAS, downstream analyses, and visual interfaces for online genome browsers. With an application to the North American Rheumatoid Arthritis Consortium (NARAC) dataset we show how the MIG algorithms can support genome-wide haplotype association studies. The MIG^{++} algorithm was already adopted by recent version of PLINK, which is one of the most popular software applications for genetic association studies worldwide.

Acknowledgements

I am very grateful to my supervisors Cristian Pattaro and Johann Gamper for their help and support during this work. Cristian Pattaro guided me in the areas of genetic association studies and statistics, while Johann Gamper helped me in the various aspects of computer science. Their iterations and comments significantly improved quality of all materials published and presented during these years. I believe that I obtained from them many professional skills which a good researcher should have.

I would like to thank my friends Francesca Pavani, Luisa Foco, Novella Carpanese, Roberto Melotti, Martin Gögele, Fabiola Del Greco, Cosetta Minelli, Damia Noce, Aude Saint Pierre, Christian Fuchsberger and Yuri D'Elia for their constant support that was expressed in so many different ways. I had an amazing time with them.

Finally, I would like to thank my family.

Preface

This work represents a result of collaboration between the Faculty of Computer Science, Free University of Bozen-Bolzano, and Center for Biomedicine, European Academy of Bozen/Bolzano, Italy.

We used data from the Genetic Analysis Workshop funded by NIH grant R01 GM031575, and data that was gathered with the support of grants from the National Institutes of Health (NO1-AR-2-2263 and RO1-AR-44422), and the National Arthritis Foundation.

Contents

List of Figures

List of Tables

List of Algorithms

Table of Symbols

Symbol	Description
s_i	Single nucleotide polymorphism (SNP)
S	Chromosome $\langle s_1, \ldots, s_n \rangle$ of n SNPs
$R_{i,j}$	Region $\langle s_i, \ldots, s_j \rangle$
The MIG, MIG$^+$ and MIG^{++} algorithms	
$w(i,j)$	SNP-pair weight between s_i and s_j SNPs
$\bar{w}(i,j)$	Weight of the region $R_{i,j}$
λ	Pruning coefficient
The S-MIG^{++} algorithm	
$P_{i,j}$	Profile of the region $R_{i,j}$
$\mathsf{T}_{n \times n}$	LD matrix
$t_{i,j}$	LD matrix element
$C_{i,j}$	Cell in LD matrix
$\mathsf{M}_{k \times k}$	Sampling matrix
($\hat{\mathbb{P}}$) $\mathbb{P}$	(Estimated) Haplotype block contour
p	Probability that $\mathbb{P} \subseteq \hat{\mathbb{P}}$
$\pi_{i,j}^{(1)}$	Proportion of *strong LD* SNP pairs in $R_{i,j}$
$\pi_{i,j}^{(2)}$	Proportion of *strong EHR* SNP pairs in $R_{i,j}$
$\pi_{i,j}^{(3)}$	Proportion of *non-informative* SNP pairs in $R_{i,j}$
$\rho_{i,j}$	Ratio of *strong LD* to *informative* SNP pairs in $R_{i,j}$
k	Number of segments in the chromosomal split
σ	Sampling fraction
η	Proportion of LD matrix elements covered by $\mathbb{P}$

Chapter 1

Introduction

1.1 Motivation

The presence of block-like patterns in human genome have intrigued genetic researchers for more than ten years and had a huge impact in the field of genetic association studies. The entire human genome can be partitioned into regions, haplotype blocks, that are characterized by a low diversity of haplotypes and an excess of linkage disequilibrium (LD) between single nucleotide polymorphisms (SNPs) compared to other genomic regions. Although, in the past, haplotype blocks have been mainly used to identify tag SNPs [1, 2], a variety of other applications is possible with currently available data.

Recently, analysis of exome-chip data has shown that within-gene LD-block distribution can be informative of the gene function and of the possible relationship between genes and specific groups of phenotypes [3]. Another application is the genome-wide haplotype association scan, which was successful in uncovering risk loci for coronary artery disease [4], Alzheimer's disease [5] and breast cancer [6]. So far, genome-wide haplotype association scans have been mostly performed based on fixed- or variable-width sliding window methods, which systematically miss haplotypes that are longer than some pre-specified maximal window width. An efficient genome-wide haplotype block recognition could help overcome such limitations, thus enhancing the biological interpretation of the results. In the study of rare variants, where collapsing methods (mostly based on gene boundaries) are becoming increasingly popular [7], the availability of haplotype blocks at genome-wide level would allow collapsing variants based on block boundaries, capturing inter-genic variants, and avoiding the problem to define the gene boundaries. Additional applications include downstream analyses of GWAS, such as pathway-based approaches, where statistics for multiple SNPs are summarized into gene-specific P-values, which are then employed for gene ranking [8]. In pathway-based

analyses, SNP-to-gene mapping is typically based on SNP proximity to the gene boundaries. With this method, when a region is gene-dense, it may be problematic to assign SNPs to a single, specific gene. An LD-based assignment would overcome this limitation and increase the power of downstream analyses [9]. In general, ignoring the LD structure in downstream analyses of GWAS can result in the misinterpretation of the findings [10].

Popular genome browsers, such as the Ensembl [11] or UCSC [12], are suitable for visualizing the LD distribution over regions of interest. However, they only allow pairwise LD calculation between markers at $<$500 kb distance from each other and do not provide any LD-block partition. With no predefined block partition, the visual assessment of such LD patterns might be influenced by investigator's subjectivity. On the other hand, the 500 kb distance constraint may limit the investigation of larger strong LD regions. With the availability of pre-calculated, threshold-free LD blocks, we would overcome both these limitations.

The most widely used haplotype block recognition algorithm is based on LD between SNPs. It was introduced by Gabriel *et al.* [13] and is implemented in many popular software applications for genetic association studies such as Haploview [14] and PLINK [15]. However, its runtime and memory usage grow quadratically with the number of SNPs restricting its possible applications only to genomic regions with up to several thousands SNPs. This problem of poor scalability becomes especially relevant for the very dense data generated by new sequencing and genotyping technologies, where even relatively short genomic region may contain thousands SNPs and the total number of SNPs in the largest chromosome reaches several millions. The focus of the thesis is the scalable implementation of the Gabriel's haplotype block partitioning algorithm that can be exploited to genome-wide scale and integrated into current routines for analysis and exploration of genetic data.

1.2 Contributions

In this thesis, we provide a series of new algorithms that significantly improve the runtime and memory usage of the Gabriel's LD-based haplotype block partitioning method. The proposed algorithms allow us to scale the haplotype block partitioning to genome-wide extent and to process very dense datasets without any restrictions on the long range LD between SNPs, which was previously impossible. Precisely, our contribution consists of the following gradual optimizations:

1. The MIG algorithm improves the memory complexity from quadratic to linear. In MIG, every genomic region has a weight that determines if this region is a haplotype block. The weight of any extended region can be computed from the weight of the previous region. This allows

constructing all possible haplotype blocks incrementally while storing only linearly growing number of weights at any computational step.

2. The MIG^{+} and MIG^{++} algorithms introduce a sophisticated search space pruning that improved the runtime by >80%. At every step of the incremental haplotype block construction in MIG, additional information about LD between SNPs in already processed area of a chromosome is obtained and collected. In MIG^{+} and MIG^{++}, we use this information to check if a current region can be further extended without violating a haplotype block definition. If not, then such region and any of its further extensions are omitted from computations.

3. The S-MIG^{++} algorithm is a sampling-based approach that outperforms MIG^{++} by another order of magnitude in terms of runtime. In S-MIG^{++}, the search space is pruned in two steps: (1) the upper limits for all possible haplotype block boundaries are estimated using only a small sample of SNP pairs; (2) exact haplotype block boundaries are refined taking into account only SNP pairs within their estimated upper limits.

1.3 Publications

The MIG, MIG^{+} and MIG^{++} algorithms were published in paper 2 and were presented at conferences 3 and 4, while the S-MIG^{++} algorithm was described in paper 1:

1. D. Taliun, J. Gamper, U. Leser and C. Pattaro. Fast Sampling-based Whole-Genome Haplotype Blocks Recognition. Under review at *ACM Transactions on Computational Biology and Bioinformatics*, 2014.

2. D. Taliun, J. Gamper and C. Pattaro. Efficient haplotype block recognition of very long and dense genetic sequences. *BMC Bioinformatics*, 15:10, 2014.

3. D. Taliun, J. Gamper and C. Pattaro. A Fast Whole-Genome Detection of LD-based Haplotype Blocks. Poster at *12th European Conference on Computational Biology*, Berlin, Germany, 2013.

4. D. Taliun, J. Gamper and C. Pattaro. Memory and Time Efficient Genome-Wide Haplotype Block Estimation. Poster at *European Mathematical Genetics Meeting*, Leiden, Netherlands, 2013.

1.4 Organization of the Thesis

The remainder of this thesis is organized as follows. In Chapter 2, we give a short introduction into genetic terminology, describe the concept of linkage disequilibrium and its measures, provide a detailed description of Gabriel's haplotype block partitioning approach. In Chapter 3, we give an overview of the existing alternative haplotype block definitions and corresponding partitioning algorithms. Chapter 4 presents the new memory and time efficient incremental computation of haplotype blocks implemented in the MIG, MIG^{+} and MIG^{++} algorithms. The impact of an approximated LD variance estimator on the runtime and haplotype block partitions is evaluated. Chapter 5 describes the new sampling-based haplotype block recognition algorithm, S-MIG^{++}, which avoids approximations. In Chapter 6, we provide an example of genome-wide haplotype block association scan in North American Rheumatoid Arthritis Consortium (NARAC) dataset. Chapter 7 presents the conclusions of the work and the future research directions.

Chapter 2

Background

In this chapter, we provide the necessary background on the linkage-disequilibrium patterns and underlying haplotype blocks that are present across the entire genome. We give a formal Gabriel's haplotype block definition and provide a detailed description of its most prominent implementation in the Haploview [14] software.

2.1 Single Nucleotide Polymorphisms

The genetic information is carried by *deoxyribonucleic acid* (DNA) that is typically represented as a sequence of four nucleotide bases: *adenine* (A), *thymine* (T), *guanine* (G), and *cytosine* (C). The DNA molecules are organized into *chromosomes*, where one chromosome corresponds to a single extremely long molecule. Every normal human cell, except for eggs and sperm, has two versions of each chromosome. Hence, it contains 23 pairs of chromosomes, where 22 pairs are *autosomes* and one pair is the sex chromosomes that determine gender. The entire set of 46 chromosomes is called the *genome*.

A particular DNA region that has two or more variants among individuals in a population is called a *polymorphism*. Every different variant that the polymorphism may obtain is called an *allele*. Several types of polymorphisms exist and the most common of them is the *single nucleotide polymorphism* (SNP). This particular type of polymorphisms corresponds to a change of a nucleotide base at a single DNA sequence position. A SNP can have from two to four alleles, although the major part of known SNPs have only two alleles and are often referred to as *biallelic*. The specific allele makeup for the individual is called a *genotype*, while a particular combination of adjacent alleles found on a single chromosome copy of the individual is termed a *haplotype*.

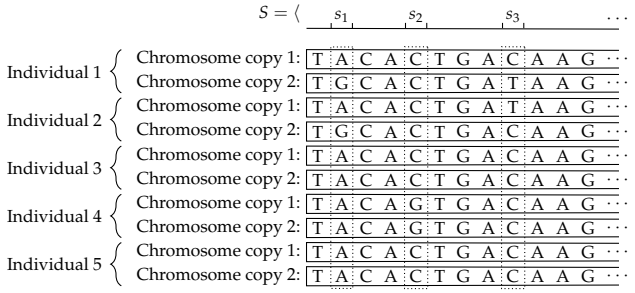


Figure 2.1: Chromosome and SNPs.

In this work, we consider biallelic SNPs in 22 human autosomes[1]. Hence, we represent a single chromosome S as a sequence $\langle s_1, \ldots, s_n \rangle$ of n SNPs, where s_i denotes a SNP at position i. A particular region in the $S = \langle s_1, \ldots, s_n \rangle$ chromosome that starts at s_i SNP and ends at s_j SNP is referred to as $R_{i,j} = \langle s_i, \ldots, s_j \rangle$. For example, Figure 2.1 illustrates three adjacent SNPs s_1, s_2, and s_3 in chromosome $S = \langle s_1, \ldots, s_n \rangle$, which was read in five individuals, yielding then ten DNA sequences (i.e., two copies of the chromosome per individual). At every SNP, two alleles were observed: A/G at s_1, C/G at s_2 and C/T at s_3. Individual 1 has A/G genotype at s_1, C/C genotype at s_2, and C/T genotype at s_3. In contrast, individual 4 has A/A genotype at s_1, G/G genotype at s_2, and C/C genotype at s_3. The ACC and GCT haplotypes are located on the 1st and 2nd chromosome copies of individual 1 at s_1, s_2, and s_3 SNPs, respectively.

2.2 Linkage Disequilibrium

The non-random association between two alleles that are found in the same haplotype is called *linkage disequilibrium* (LD). Lewontin and Kojima [16] introduced the $D_{i,j}$ measure of LD between two SNPs s_i and s_j with alleles a_1/a_2 and b_1/b_2, respectively:

$$D_{i,j} = f_{a_1b_1} \cdot f_{a_2b_2} - f_{a_1b_2} \cdot f_{a_2b_1}, \tag{2.1}$$

where $f_{a_1b_1}$, $f_{a_1b_2}$, $f_{a_2b_1}$ and $f_{a_2b_2}$ denote the relative frequencies of the four possible haplotypes a_1b_1, a_1b_2, a_2b_1 and a_2b_2, correspondingly.

Since the range of $D_{i,j}$ depends on the frequencies of alleles, Lewontin [17] later proposed a normalized LD coefficient, $D'_{i,j}$:

$$D'_{i,j} = \frac{D_{i,j}}{D_{max_{i,j}}}, \tag{2.2}$$

[1]Here and later throughout the thesis, genome refers to complete set of autosomes, and chromosome and autosome are used interchangeably.

where

$$D_{max_{i,j}} = \begin{cases} \min(f_{a_1} \cdot f_{b_2}, f_{a_2} \cdot f_{b_1}) & \text{if } D_{i,j} > 0 \\ \min(f_{a_1} \cdot f_{b_1}, f_{b_1} \cdot f_{b_2}) & \text{if } D_{i,j} < 0 \end{cases}$$

and f_{a_1}, f_{a_2}, f_{b_1} and f_{b_2} are the marginal frequencies of the corresponding alleles a_1, a_2, b_1 and b_2 in the two SNPs. The $D'_{i,j}$ LD coefficient goes from -1 to $+1$ and is independent of the allelic frequencies of the two SNPs involved. When $D'_{i,j} = 0$, the two SNPs are independent (perfect equilibrium), while $|D'_{i,j}| = 1$ indicates that no more than three of the four possible haplotypes are being observed in the sample (complete disequilibrium).

Figure 2.2 shows $D'_{1,2}$, $D'_{1,3}$, and $D'_{2,3}$ between s_1, s_2, and s_3 SNPs. For example, the marginal frequencies of alleles A and G at s_1 are $f_A = 0.8$ and $f_G = 0.2$, while the marginal frequencies of alleles C and T at s_3 are $f_C = 0.8$ and $f_T = 0.2$. The relative frequencies of four possible haplotypes AC, AT, GC, and GT at s_1 and s_3 are $f_{AC} = 0.7$, $f_{AT} = 0.1$, $f_{GC} = 0.1$ and $f_{GT} = 0.1$. This results in $D_{1,3}$ equal to

$$D_{1,3} = 0.7 \cdot 0.1 - 0.1 \cdot 0.1 = 0.06.$$

Since $D_{1,3} > 0$, then

$$D_{max_{1,3}} = \min(0.8 \cdot 0.2, 0.2 \cdot 0.8) = 0.16.$$

Consequently, the normalized $D'_{1,3}$ LD coefficient between s_1 and s_3 SNPs is equal to

$$D'_{1,3} = \frac{0.06}{0.16} = 0.375.$$

The remaining normalized LD coefficients $D'_{1,2}$ and $D'_{2,3}$ are computed in a similar way. Note, that $D'_{i,j}$ is symmetric but not transitive.

Hill and Robertson [18] proposed an alternative normalized LD coefficient, $r^2_{i,j}$, that is currently more commonly used than $D'_{i,j}$ to identify independent signals in genome-wide association studies (GWASs):

$$r^2_{i,j} = \frac{D^2_{i,j}}{f_{a_1} \cdot f_{a_2} \cdot f_{b_1} \cdot f_{b_2}}. \tag{2.3}$$

However, it has been shown that $r^2_{i,j}$ is not significantly more precise, accurate or efficient than $D'_{i,j}$ [19]. Both coefficients capture similar information but their range of variation can be very different. In contrast to $D'_{i,j}$, the range of $r_{i,j}$ greatly depends on the allele frequencies and equals -1 or $+1$ only when the two SNPs have the same allele frequency. In such cases, $|r_{i,j}| = 1$ indicates that knowing the allele at one SNP allows determining the allele at the other SNP (perfect disequilibrium). But when the two SNPs have very different allele frequencies, the interpretation of $r^2_{i,j}$ becomes difficult. This is especially relevant with the data generated by the

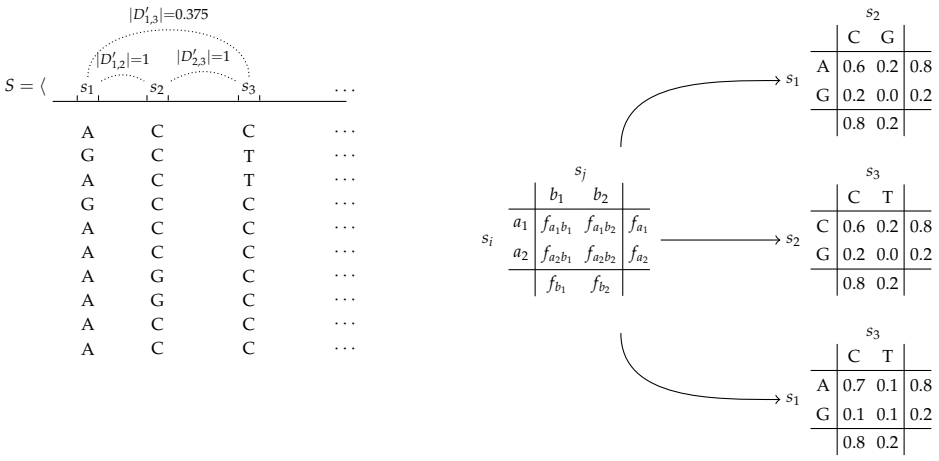


Figure 2.2: The $D'_{i,j}$ linkage disequilibrium coefficient.

new sequencing technologies that allow genotyping SNPs over a very wide spectrum of allele frequencies. In such situations, the $r^2_{i,j}$ may fail to identify the correct relationship between nearby variants. In GWAS, this may lead to a wrong definition of the identified loci. Therefore, $D'_{i,j}$ should remain the statistics of choice for LD modeling because of its more direct biological interpretation. It reflects the history of recombination, mutation, and selection events that cause some chromosomal regions to be less diverse than others and, therefore, influence the haplotype distribution.

2.3 Alternative Ways to Model D' Distribution

In this work, we considered two different methods to estimate the confidence interval (CI) of $|D'_{i,j}|$. First method is the likelihood-based procedure proposed by Wall and Pritchard [20] and implemented in Haploview [14], which requires from 100 to 1,000 iterations. The second method is based on an approximated estimator of the $D'_{i,j}$ variance, as proposed by Zapata *et al*. [21]. The latter method is computationally cheaper.

2.3.1 The Wall and Pritchard (WP) Method

The true allele frequencies of each SNP are assumed to be equal to the observed allele frequencies. The likelihood of the data in the four-fold table obtained by crossing any SNP pair, conditional to the $|D'_{i,j}|$ value, can be expressed as $l{=}P(data||D'_{i,j}|)$. l is evaluated at each value of $|D'_{i,j}| = 0.001 \times p$, with $p = 0, 1, \ldots, 1000$. The lower bound of CI is defined as the largest value of $|D'_{i,j}|$ such that $\sum_{k=0}^{p-1} l(k) / \sum_{k=0}^{1000} l(k) \leq \alpha$, where α is the signif-

icance level. Similarly, the upper bound of CI is defined as the smallest value of $|D'_{i,j}|$ such that $\sum_{k=p+1}^{1000} l(k) / \sum_{k=0}^{1000} l(k) \leq \alpha$.

2.3.2 The Approximate Variance (AV) Method

Consider two SNPs, s_i and s_j, with alleles a_1/a_2 and b_1/b_2, respectively. Zapata *et al.* [21] showed that the variance of $D'_{i,j}$ can be approximated as follows:

$$V(D'_{i,j}) \approx \Big(\big(1 - |D'_{i,j}|\big) \times \big(N \cdot V(D_{i,j}) - |D'_{i,j}| D_{max_{i,j}} (f_{a_1} f_1 + f_{a_2} f_2 - 2|D_{i,j}|)\big) + |D'_{i,j}| f_3 (1 - f_3) \Big) \Big/ \Big(N \cdot D^2_{max_{i,j}} \Big),$$

where N is the total number of observed haplotypes; f_1 is f_{b_1} when $D'_{i,j} > 0$ or f_{b_2} when $D'_{i,j} < 0$; f_2 is f_{b_2} when $D'_{i,j} > 0$ or f_{b_1} when $D'_{i,j} < 0$; f_3 is $f_{a_1 b_1}$, $f_{a_1 b_2}$, $f_{a_2 b_1}$, and $f_{a_2 b_2}$ when $D_{max_{i,j}}$ is $f_{a_1} f_{b_1}$, $f_{a_1} f_{b_2}$, $f_{a_2} f_{b_1}$, and $f_{a_2} f_{b_2}$, respectively; and

$$V(D_{i,j}) \approx \big(f_{a_1} f_{a_2} f_{b_1} f_{b_2} + D_{i,j}(f_{a_2} - f_{a_1})(f_{b_2} - f_{b_1}) - D^2_{i,j}\big)/N.$$

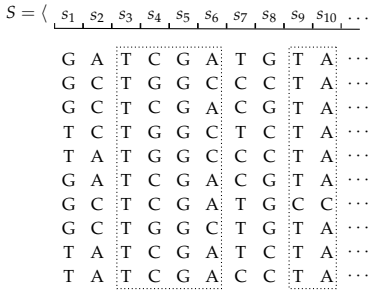
When $D'_{i,j} = \pm 1$, then $V(D'_{i,j}) = 0$. The $1 - \alpha$ CI of $D'_{i,j}$ is equal to $D'_{i,j} \pm Z_{\alpha/2} \sqrt{V(D'_{i,j})}$, where $Z_{\alpha/2}$ is the $1 - \alpha/2$ percentile of the standard normal distribution.

2.4 Haplotype Blocks

The maximal theoretically possible number of different haplotypes at m adjacent biallelic SNPs is equal to 2^m. However, the genetic make-up of different individuals at some adjacent SNPs can be almost identical and at such genomic regions the number of different haplotypes is significantly less than expected. Such regions with reduced haplotype diversity were termed *haplotype blocks* [1, 22].

For example, consider two adjacent SNPs s_1 and s_2 on Figure 2.3. Since all four theoretically possible haplotypes GA, GC, TC and TA were observed, the $R_{1,2} = \langle s_1, s_2 \rangle$ region has high haplotype diversity and is a poor candidate for haplotype block. In contrast, at adjacent SNPs s_3, s_4, s_5, and s_6 only two out of $2^4 = 16$ theoretically possible haplotypes were observed: TCGA, TGGC. Thus, the $R_{3,6} = \langle s_3, \ldots, s_6 \rangle$ region may be a good candidate for haplotype block.

There are many biological factors that influences the DNA diversity, but the main factor for haplotype blocks is considered to be *recombination* events. By analyzing haplotype structure on chromosome 5q31, Daly *et al.* [22] observed that regions of reduced haplotype diversity were



SNPs	Diversity	
	Haplotype	Frequency
s_1, s_2	GA	2
	GC	4
	TC	1
	TA	3
s_3, s_4, s_5, s_6	TCGA	6
	TGGC	4
s_7, s_8	TG	3
	TC	2
	CG	2
	CC	3
s_9, s_{10}	TA	9
	CC	1

Figure 2.3: Regions of reduced haplotype diversity.

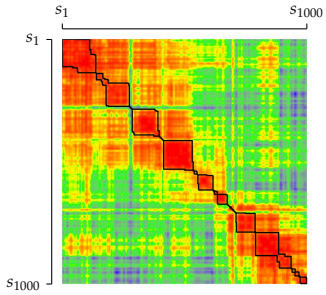


Figure 2.4: LD heatmap of chr20:14,759,169-15,028,962 in the 1000 Genomes Project data.

punctuated by apparent sites of recombination. Moreover, the presence of underlying haplotypes of limited diversity is reflected in LD patterns. Using SNPs across entire chromosome 22, Dawson *et al.* [23] observed regions of high LD that are interspersed with regions of low LD, where high LD was in regions of low recombination. Figure 2.4 illustrates such LD pattern of chr20:14,759,169-15,028,962 in the 1000 Genomes Project data [24]. Red, green, and blue colors correspond to high, moderate, and low $|D'_{i,j}|$, respectively. The black line outlines the haplotype blocks. Hence, the LD between two SNPs is related to the distance between them in the genome: nearby SNPs are more likely to be in high LD than distant SNPs [25].

2.5 Gabriel's Haplotype Blocks Recognition

The most commonly used haplotype block definition was proposed by Gabriel *et al.* [13] and exploits the interconnection between underlying haplotypes and LD patterns. The SNP pairs are classified into three classes based on 90% CI of the $|D'_{i,j}|$ LD coefficient.

Definition 1. (*Classification of SNP Pairs*). Let $S = \langle s_1, \ldots, s_n \rangle$ be a chromo-

some and $[lb_{i,j}, ub_{i,j}] \subseteq [0,1]$ be the 90% confidence interval of the $|D'_{i,j}|$ LD coefficient between s_i and s_j. The SNPs s_i and s_j form

- a *strong LD* pair if $lb_{i,j} \geq 0.7$ and $ub_{i,j} \geq 0.98$,
- a *strong EHR* (evidence of historical recombination) pair if $ub_{i,j} < 0.9$, and
- a *non-informative* pair otherwise.

The *strong LD* and *strong EHR* pairs are also referred to as *informative* pairs. Based on this classification, Gabriel *et al.* [13] introduced the following formal definition of haplotype blocks.

Definition 2. (*Haplotype Block*). Let $R_{i,j} = \langle s_i, \ldots, s_j \rangle$ be a region of consecutive SNPs in a chromosome $S = \langle s_1, \ldots, s_n \rangle$, l the number of *strong LD* SNP pairs in $R_{i,j}$, and r the number of *strong EHR* SNP pairs in $R_{i,j}$. Then, $R_{i,j}$ is a *haplotype block* if

(a) the two outermost SNPs, s_i and s_j, form a *strong LD* pair, and

(b) there is at least a proportion d of *informative* pairs that are *strong LD* pairs, i.e.: $l/(l+r) \geq d$.

In their original work, Gabriel *et al.* [13] set $d = 0.95$ after investigating the fractions of *strong LD* SNP pairs in genomic regions of different length and in different populations[2].

Definition 2 implies two important properties: haplotype blocks may partially or completely overlap each other, and a haplotype block may contain (sub)regions that do not represent haplotype blocks, i.e., do not satisfy the above definition.

The most prominent implementation of Definition 2 is shipped with the popular tool Haploview [14] and works in two steps[3]: (1) all regions satisfying Definition 2(a) are collected in a set of candidate haplotype blocks; (2) from this set of candidates, a subset of non-overlapping regions that satisfy Definition 2(b) is selected. In the first step, the entire chromosome is scanned and, for every SNP pair, the $|D'_{i,j}|$ CI is computed and stored in an $n \times n$ matrix. The matrix is then traversed to identify the pairs that satisfy Definition 2(a). These pairs mark regions of different length that are candidates to become haplotype blocks. In the second step, the candidate regions are sorted by decreasing length and processed starting with the largest one. If a region satisfies Definition 2(b), it is classified as a haplotype block, and all other overlapping candidate regions are discarded.

[2]Here and later throughout the thesis, if not stated otherwise, we consider d always equal to 0.95.

[3]Here and later throughout the thesis, the Haploview algorithm and the Gabriel's algorithm are used interchangeably.

Regions not satisfying Definition 2(b) are skipped. This process continues with the next largest candidate region, until the candidate set is completely processed and the list of haplotype blocks is complete.

The overall complexity of the algorithm is mainly determined by the first step. More specifically, the $\Theta(n^2)$ time and memory complexity is due to the computation and maintenance of the $n \times n$ CI matrix. For this reason, we focused our improvements on the first step of the algorithm.

Chapter 3

Related Work

The haplotype blocks have received considerable attention in the research community, leading to a multitude of different haplotype block definitions and algorithms for their recognition. However, in the absence of a commonly accepted formal definition, the most widely used definition is that proposed by Gabriel *et al.* [13] and implemented in popular software applications such as Haploview [14] and PLINK [15]. In this chapter, we provide a short overview of other available haplotype block definitions and algorithms for their recognition.

3.1 Recombination Crossovers

Wang *et al.* [26] defined a haplotype block as a region of consecutive SNPs where no recombination crossover was observed between any two SNPs in the region. The presence of the recombination crossover is tested using the four-gamete test (FGT): if all four possible haplotypes were observed in two biallelic SNPs, then at least one historical recombination event has happened. Hence, FGT implies a very conservative definition of haplotype blocks. The proposed algorithm for haplotype blocks recognition starts constructing a block from the very left of the chromosome by appending SNPs while FGT didn't show recombination with any other SNP in the block. If the FGT showed the recombination, then the current block ends and the new block starts. The algorithm has $O(h \cdot n)$ runtime complexity, where h is the number of SNPs in the longest haplotype block and n is the total number of SNPs in a chromosome. Although the computationally fast FGT makes this algorithm very scalable, it didn't become as popular as other methods.

3.2 Haplotype Coverage

Patil *et al.* [1] proposed the haplotype block definition that uses a haplotype coverage criteria: a region of consecutive SNPs is a haplotype block if more than 80% of haplotypes constitute of *common* haplotypes represented more than once in the region (i.e. 80% coverage). For every haplotype block, exists a minimal number of *tag* SNPs that uniquely discriminate common haplotype in the block. The proposed greedy algorithm for haplotype blocks recognition maximizes the length of the blocks and minimizes the number of required *tag* SNPs. It works in two steps: (1) all possible regions of one SNP and longer are considered and regions with less than 80% coverage are excluded; (2) the overlapping regions are excluded prioritizing longer regions with less *tag* SNPs. The first step leads to exponential runtime complexity and makes this greedy algorithm not scalable.

Zhang *et al.* [27] introduced a dynamic programming algorithm using the coverage-based haplotype block definition. Assuming the precomputed tag SNPs for every haplotype block, the space and runtime complexities of the proposed algorithm are, correspondingly, $O(m \cdot n)$ and $O(h \cdot m \cdot n)$, where m is the number of tag SNPs, h is the number of SNPs in the longest haplotype block, and n is the total number of SNPs in chromosome. However, as Zhang *et al.* [27] noted, the computation of minimal number of tag SNPs in the block is NP complete. Later, Zhang *et al.* [2, 28] and Chen *et al.* [29] implemented a set of similar dynamic programming algorithms under various constraints.

3.3 Minimal Description Length

Anderson and Novembre [30] proposed an algorithm for haplotype blocks recognition based on the minimal description length (MDL) principle, which captures both the LD decay and the reduced haplotype diversity. According to the MDL principle, the best set of adjacent non-overlapping haplotype blocks is the set that minimizes the number of bits required to encode the data i.e., minimizes the description length of the data. The description length for a set of haplotype blocks is computed based on the estimated probabilities of haplotypes in blocks using Hidden Markov Chain models. The iterative dynamic programming (IDP) algorithm explores all possible sets of haplotype blocks and selects one that minimizes the description length. It has $O(n^3)$ runtime complexity and $O(n^2)$ memory complexity, where n is the total number of SNPs in chromosome. The alternative approximate algorithm, IADP, has $O(n^2)$ runtime complexity and $O(n)$ memory complexity.

Manilla et al. [31] proposed an alternative MDL-based algorithm for haplotype block partitioning. In contrast to Anderson and Novembre [30],

the description length for every haplotype block is computed without employing Markov model. Instead, haplotypes inside a block are clustered using k-means clustering and the description length depends on the number of clusters within the block and how closely haplotypes within blocks cluster together. The developed dynamic programming algorithm for computing an optimal block partition has $O(s \cdot n^3)$ time complexity, where s is the number of haplotypes (i.e. 2 $\times$ number of individuals) and n is the total number of SNPs in chromosome.

Greenspan and Geiger [32] developed another MDL-based algorithm based on Markov model. Their model is more complex than the model proposed by Anderson and Novembre [30]. It allows accounting for recombination hotspots, bottlenecks, genetic drifts, and mutations. In addition to hapotype data, it accepts genotype data and performs haplotype resolution. However, Greenspan and Geiger [32] do not use dynamic programming to infer a single globally optimal partition as in the method by Anderson and Novembre [30], but infer an ensemble of locally optimal models to allow for the ambiguity of block partitioning. Authors state that if a bound is placed on the maximum number of SNPs and ancestors in any block, then the time complexity of their algorithm is $O(m \cdot s \cdot n)$, where m is the number of models to be sampled (e.g. 100 or more), s is the number of haplotypes and n is the total number of SNPs in chromosome.

3.4 Probabilistic Methods

Pattaro *et al.* [33] introduced the MCMC Algorithm To Identify blocks of Linkage DisEquilibrium (MATILDE) for clustering consecutive SNPs. The MATILDE algorithm assumes that pairwise LD statistics arises from one of two separate probability distribution functions, where one is the LD distribution and another is the independence (i.e. non-LD) distribution. Using the two distributions, algorithm constructs the vector of probability scores, $\underline{\gamma}$, for every block boundary between two adjacent SNPs s_i and s_{i+1}. MATILDE allows using $D'_{i,j}$ or $r^2_{i,j}$ as an LD statistics of choice. The non-LD probability distribution is estimated non-parametrically by randomly permuting genotypes between individuals, while the LD probability distribution is assumed to follow Beta distribution with the unknown parameters $\alpha > 0$ and $\beta > 0$ such that $\beta > \alpha$. Then, for two given distributions, the log-likelihood function is expressed using three unknown parameters $\underline{\gamma}$, α and β, which are estimated with Metropolis-Hastings algorithm.

The probabilistic graphical models (PAMs) offer a framework for an accurate modeling of dependencies between random variables using graphs. PAMs were considered as a promising approach for LD modeling and for clustering SNPs [34]. They allow an accurate identification of SNP clusters even in situations when SNPs are not necessary contiguous. The LD may be

modeled using directed graphs (Bayesian networks) or undirected graphs (Markov random fields). Mourad *et al.* [35] proposed to use a forest of hierarchical latent class models (FHLCM) which consists of a directed acyclic graph whose non-connected components are trees. In FHLCM, leaves (observed variables) represent SNPs and internal nodes are latent (unobserved variables) organized in multiple layers, while edges represent conditional dependencies between variables. The set of parameters in FHLCM is a matrix of all conditional probability distributions. The learning of FHLCMs is computationally demanding and consists of two parts: structure (i.e. nodes and edges) learning, learning of parameters. The corresponding CFHLC algorithm proposed by Mourad *et al.* [35] is able to process only up to ten thousand of SNPs.

3.5 Comparison of Existing Methods

Schwartz *et al.* [36] compared the haplotype block partitions derived by recombination-based Wang's [26] method, LD-based Gabriel's [13] method, and haplotype coverage-based Patil's [1] method. The assessment was performed using 22,047 SNPs in chromosome 21 and 88 SNPs in chromosome 8. The measure of similarity between partitions was computed using the number of shared haplotype block boundaries. The experimental results showed very poor agreement between haplotype block partitions derived by different methods. However, the number of shared block boundaries was much greater than can be expected by chance. In terms of the similarity between partitions, the recombination-based method was much closer to LD-based and haplotype coverage-based methods than either of those was to the other. Schwartz *et al.* [36] suggested that the contradiction in block boundaries is due to the fact that haplotype blocks are not sharply defined as the concept of discrete blocks would imply.

Schulze *et al.* [37] performed a formal comparison between Gabriel's and Patil's methods using 33 SNPs in chromosome 18q21.32-33, 55 SNPs in chromosome 22q13.31-32 and 54 SNPs in chromosome 22q13.33. Their results showed that haplotype coverage-based method consistently identified fewer and larger haplotype blocks than the LD-based method. In both methods, the number of identified haplotype blocks decreased when rarer SNPs were excluded. Schulze *et al.* [37] noted that block partitions are sensitive to different parameter choices (i.e. haplotype coverage and $|D'_{i,j}|$ CI thresholds), but the results could not be reconciled by adjustment of this parameters.

Ding *et al.* [38] compared Wang's, Gabriel's and Patil's methods using systematic simulations under various population-genetics parameters (mutation and recombination rates) and under different recombination models (coalescent model with uniform recombination and recombination

hotspots). They concluded that the recombination-based method appears to be much closer to the LD-based method than either of those is to the haplotype coverage-based method under two recombination models.

Using coalescent simulations Indap *et al.* [39] generated 1,000 haplotypes with high SNPs density (1,349 SNPs in 200kb region) to compare haplotype block partitions derived by Gabriel's method, Patil's method and MDL-based method from Anderson and Novembre [30]. Their results showed that the haplotype coverage-based method generally inferred the largest number of blocks of smallest size, which contradicts to Schulze *et al.* [37]. The MDL-based method inferred the fewest number of blocks of largest size. While there were very few exact matching block boundaries between different partitions, there were a large amount of common block regions between them. For all methods, the number of detected haplotype blocks increased with the increased density of SNPs. Consequently, the average number of base pairs per block decreased with a higher density of SNPs. The increase in SNPs density had a more dramatic effect on the Gabriel's method than other two methods due to fact that LD patterns are sensitive to SNPs density and may change with the addition of more SNPs. In contrast to Schulze *et al.* [37], the number of blocks detected by each method didn't decrease when rare SNPs were excluded. Indap *et al.* [39] concluded that there is a great divergence in haplotype blocks detected by each method and advised to use multiple algorithms in parallel to comprehensively account for haplotype block structure in genetic studies.

Pattaro *et al.* [33] proposed the probabilistic MATILDE algorithm for haplotype block partitioning and compared it to Wang's, Gabriel's and Patil's methods using first 500 SNPs in chromosome 14q11 from HapMap phase II [40] dataset. Methods were compared using Cohen's κ statistics on the number of shared break points. The results showed that the haplotype block partition derived by MATILDE was more similar to the LD-based method than to the haplotype coverage-based method, which can be explained by the fact that block estimations in MATILDE are based on LD patterns. Also, experiments confirmed results of Schulze *et al.* [37] that Gabriel's method generate higher number of smaller blocks compared to Patil's method.

In summary, all evaluations agree on existence of block-like patterns in human genome. Although the agreement between different block partitioning algorithms is low, it is still higher than can be expected by chance. The differences is due to the absence of a universal haplotype block definition, since it is not clear how exactly genetic processes of mutation, drift, and recombination affect the formation of haplotype blocks. Despite this, the concept of haplotype blocks has a big impact on analysis and interpretation of genetic data.

Chapter 4

Incremental Computation of Haplotype Blocks

In this chapter, we describe how we improved the efficiency and scalability of Haploview algorithm by adopting an incremental computation of the haplotype blocks based on iterative chromosome scans. The incremental computation strategy led to an algorithm, termed MIG^{++}, that has $\Theta(n)$ memory complexity and omits more than 80% of the pairwise LD computations, while obtaining exactly the same final haplotype bock partition as Haploview. In contrast to Haploview, the new algorithm can consider pairwise LD between SNPs at any distance.

4.1 SNP-pair and Region Weights

The core ideas of our optimizations are to compute haplotype blocks incrementally and to omit, as soon as possible, regions that cannot be extended to larger blocks due to an insufficient proportion of *strong LD* SNP pairs. In this way, we avoid both unnecessary computations and the storage of an $n \times n$ CI matrix. The incremental haplotype block computation is based on the concepts of a *SNP-pair weight* and a *region weight* described below.

Definition 3. (*SNP-pair weight*). Let d be a threshold as defined in Definition 2. For a given pair of SNPs s_i and s_j, the SNP-pair weight, $w(i,j)$, is defined as follows:

$$w(i,j) = \begin{cases} 1-d & \text{if } s_i \text{ and } s_j \text{ is a } \textit{strong LD} \text{ pair,} \\ -d & \text{if } s_i \text{ and } s_j \text{ is a } \textit{strong EHR} \text{ pair,} \\ 0 & \text{otherwise.} \end{cases}$$

Definition 4. (*Region weight*). Let $R_{i,j}$ be a chromosome region. The region

weight of $R_{i,j}$, $\bar{w}(i,j)$, is defined as the sum of all SNP-pair weights in $R_{i,j}$:

$$\bar{w}(i,j) = \sum_{v=i+1}^{j} \sum_{u=i}^{v} w(u,v).$$

The following theorem defines a haplotype block based on the region weight.

Theorem 1 *Let $R_{i,j}$ be a chromosome region. $R_{i,j}$ is a haplotype block if $w(i,j) = 1-d$ and $\bar{w}(i,j) \geq 0$.*

Proof. From Definition 3, if SNPs s_i and s_j form a *strong LD* pair, then $w(i,j) = 1-d$. Therefore, Definition 2(a) is satisfied. $R_{i,j}$ contains $L = \sum_{v=i+1}^{j} \sum_{u=i}^{v} 1$ possible SNP pairs, of which l are *strong LD* pairs, r are *strong EHR* pairs, and the remaining ones are *non-informative*. From Definitions 3 and 4, it follows that $\bar{w}(i,j) = \sum_{v=i+1}^{j} \sum_{u=i}^{v} w(u,v) = l(1-d) + r(-d) + (L-l-r)\cdot 0 = l - d(l+r)$. If $\bar{w}(i,j) \geq 0$, then $l - d(l+r) \geq 0 \implies l/(l+r) \geq d$. Therefore, Definition 2(b) is also satisfied. ■

Theorem 1 is the basis for the incremental haplotype block reconstruction, which is the core of our optimizations. In the following, we present three gradual improvements of the Haploview algorithm: a memory-efficient implementation based on the Gabriel *et al.* [13] definition (MIG); MIG with additional search space pruning (MIG$^+$); and MIG$^+$ with iterative chromosomal processing (MIG^{++}). Theorem 1 ensures that all three algorithms produce block partitions that are identical to the original Haploview results.

4.2 The MIG Algorithm

For a given chromosome S containing n SNPs, the maintenance of an $n \times n$ matrix containing all the $|D'_{i,j}|$ CIs can be avoided by storing n region weights in a unidimensional vector $W_{n\times 1}$. In each element of W, $W[i]$, we store the weight of a chromosomal region that starts at SNP s_i. When the region is enlarged by including additional SNPs to the right of s_i, the weight $W[i]$ is updated accordingly. This procedure, illustrated in Figure 4.1, begins with setting all the weights to 0. At the initial stage, the vector W represents all one-SNP regions. Then, the region starting at SNP s_1 is enlarged by including the next SNP, s_2. Therefore, starting from s_2, chromosome S is processed one SNP after the other, from left to right. For a SNP s_j, with $j \geq 2$, all SNP pair weights $w(i,j)$, $i = j-1, ..., 1$, are computed and added up as $\sigma = w(j-1,j) + \cdots + w(i,j)$.

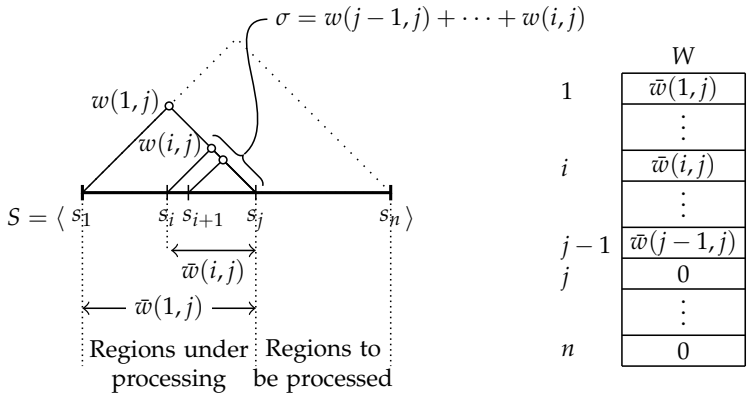


Figure 4.1: Processing a chromosome with the MIG algorithm.

σ and $W[i]$ are updated for every computed weight $w(i,j)$. Before the update, $\sigma = w(j-1,j) + \cdots + w(i-1,j)$ and $W[i]$ contains the region weight $\bar{w}(i,j-1)$, which was already computed for the previous SNP s_{j-1}. Then, σ is incremented by $w(i,j)$ and $W[i]$ is incremented by the new value of σ. $W[i]$ now represents the region weight $\bar{w}(i,j)$, i.e., $\bar{w}(i,j) = \bar{w}(i,j-1) + w(j-1,j) + \cdots + w(i,j)$. Whenever $w(i,j) = 1-d$ and $\bar{w}(i,j) \geq 0$, Theorem 1 is satisfied and the region $\langle s_i, \ldots, s_j \rangle$ is added to the set of candidate haplotype blocks. This procedure is repeated with the next SNP, s_{j+1}. The pseudocode is provided in Algorithm 1.

Example 1. An example of the first three computational steps is given in Figure 4.2. The vector W and the variable σ are initialized to 0. The processing starts at s_2 with the analysis of the region $\langle s_1, s_2 \rangle$. The SNP-pair weight $w(1,2)$ is computed and added to σ. Then, the region weight $\bar{w}(1,2)$ is incrementally computed as $W[1] + \sigma$ and stored in $W[1]$ (by replacing the old value). Next, SNP s_3 is processed. After initializing $\sigma = 0$, weight $w(2,3)$ is computed and stored in σ. $\bar{w}(2,3)$ is incrementally determined as $W[2] + \sigma$, and stored in $W[2]$. Then, weight $w(1,3)$ is computed, and $\bar{w}(1,3) = \bar{w}(1,2) + w(2,3) + w(1,3) = W[1] + \sigma$. The next SNP to the right, s_4, is processed in a similar way.

MIG reduces the memory complexity from $\Theta(n^2)$ to $\Theta(n)$. Moreover, instead of identifying candidate regions that satisfy only Definition 2(a) (as in Haploview), MIG checks immediately both conditions (a) and (b). This yields a smaller set of candidate blocks, and therefore indirectly speeds up also the second step of the Haploview algorithm.

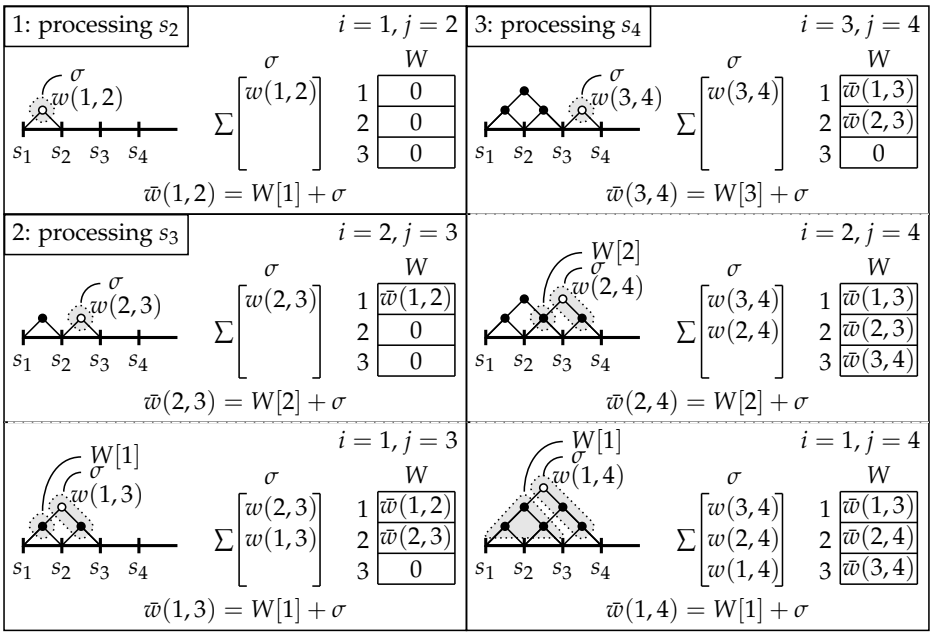


Figure 4.2: The first three computational steps of the MIG algorithm.

4.3 The MIG$^+$ Algorithm

While MIG drastically reduces the memory requirements by avoiding the maintenance of the CI matrix, it still computes weights for all SNP pairs, totaling $n(n-1)/2$ computations as in Haploview. To omit more unnecessary computations, we apply a search space pruning to the MIG algorithm to identify regions that cannot be further extended to form a haplotype block. The pseudocode of the new algorithm, MIG$^+$, is shown in Algorithm 2.

Instead of computing weights for all pairs of SNPs, only weights $w(j-1,j),\ldots,w(b,j)$ are computed, where $b = min(\{i \mid 1 \leq i < j \wedge \bar{w}_{max}(i,j) \geq 0\})$ and $\bar{w}_{max}(i,j) = max\{\bar{w}(i,k) \mid j < k \leq n\}$. The function $\bar{w}_{max}(i,j)$ is an upper bound for the weight of all regions $\langle s_i,\ldots,s_j,\ldots,s_k\rangle$ that start at s_i and end after s_j, i.e., those extending beyond the region $\langle s_i,\ldots,s_j\rangle$. If $\bar{w}_{max}(i,j) < 0$ for some i, none of the regions $\langle s_i,\ldots,s_k\rangle$ can satisfy Definition 2(b). The smallest i, that can be a potential starting point of a region with a positive weight, can therefore be set as breakpoint b. Regions starting left of b and stopping right of j receive negative weights and are discarded. Figure 4.3 schematically illustrates the pruning step. $\bar{w}_{max}(i,j)$ is computed taking into account the SNP-pair weights computed at the previous stages and colored in gray. Since $\bar{w}_{max}(i,j) < 0$ for any region that starts before the SNP s_b and ends after the SNP s_j, then the SNP-pair weights within the hatched area are omitted from computation.

Algorithm 1: MIG

Data: $S = \langle s_1, \ldots s_n \rangle$
Result: $H = \emptyset$

$W \leftarrow \langle 0, \ldots, 0 \rangle$;
for $j = 2$ **to** n **do**
    $\sigma \leftarrow 0$;
    for $i = j - 1$ **downto** 1 **do**
        $w \leftarrow w(i, j)$;
        $\sigma \leftarrow \sigma + w$;
        $W[i] \leftarrow W[i] + \sigma$;
        if $w = 1 - d$ **and** $W[i] \geq 0$ **then**
            $H \leftarrow H \cup \langle s_i, \ldots, s_j \rangle$;
return H;

Algorithm 2: MIG$^+$

Data: $S = \langle s_1, \ldots s_n \rangle$
Result: $H = \emptyset$

$W \leftarrow \langle 0, \ldots, 0 \rangle$;
$new\_b \leftarrow 1$;
for $j = 2$ **to** n **do**
    $\sigma \leftarrow 0$;
    $b \leftarrow new\_b$;
    $new\_b \leftarrow j$;
    for $i = j - 1$ **downto** b **do**
        $w \leftarrow w(i, j)$;
        $\sigma \leftarrow \sigma + w$;
        $W[i] \leftarrow W[i] + \sigma$;
        if $w = 1 - d$ **and** $W[i] \geq 0$ **then**
            $H \leftarrow H \cup \langle s_i, \ldots, s_j \rangle$;
        if $\bar{w}_{max}(i, j) \geq 0$ **then**
            $new\_b \leftarrow i$;
return H;

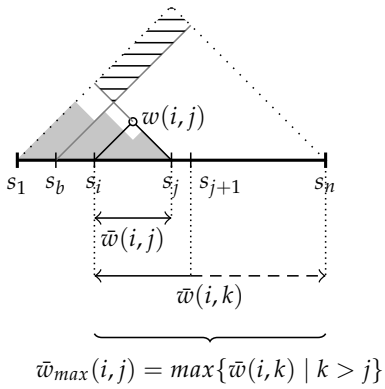


Figure 4.3: Processing a chromosome with the MIG$^+$ algorithm.

The upper bound, $\bar{w}_{max}(i,j)$, is estimated assuming that all unprocessed SNPs to the right of s_j are in *strong LD* with each other and with all SNPs in the region $\langle s_i, \ldots, s_j \rangle$. Then, $\bar{w}(i,k) \leq \bar{w}(i,j) + (1-d) \cdot L$, where $\bar{w}(i,j)$ is already computed and $L = ((j-i+1)+(k-i))(k-j)/2$ is the number of unprocessed SNP pairs. Since L is largest for the longest region $\langle s_i, \ldots, s_n \rangle$, we have $max\{\bar{w}(i,k) \mid k > j\} \leq \bar{w}(i,j) + (1-d) \cdot ((j-i+1)+(n-i))(n-j)/2$, and the estimated upper bound $\bar{w}_{max}(i,j)$ is defined as follows:

$$\bar{w}_{max}(i,j) = \bar{w}(i,j) + (1-d) \cdot ((j-i+1)+(n-i))(n-j)/2. \qquad (4.1)$$

The MIG$^+$ algorithm performs at most $\lambda n(n-1)/2$ computations, where λ, $1-d \leq \lambda \leq 1$, depends on the data. The worst case of $\lambda = 1$ occurs only in the unlikely situation when a few very large blocks span an entire chromosome.

4.4 The MIG^{++} Algorithm

A limitation of the MIG$^+$ algorithm is its blindness about the unprocessed area to the right of the current SNP s_j. Assuming *strong LD* for all SNP pairs in this area results in a conservative upper bound, $\bar{w}_{max}(i,j)$, for the region weights. An additional optimization step allows obtaining a more precise estimate of $\bar{w}_{max}(i,j)$ and further pruning of unnecessary computations. The pseudocode of the modified algorithm, MIG^{++}, is given in Algorithm 3.

The improved algorithm is an iterative procedure that, at each iteration, scans the chromosome from left to right and computes the weights only for a limited number of SNP pairs. For a SNP s_j, the SNP pairs considered in an iteration are restricted to a window of size *win*: only the weights $w(j-1,j), \ldots, w(t,j)$ are computed, where $t = \max(\{b, j-win\})$ and $1 \leq win \leq n$. At each new iteration, the window size is increased

Algorithm 3: MIG^{++}

Data: $S = \langle s_1, \dots s_n \rangle$
Input: *win*
Result: $H = \emptyset$

$W \leftarrow \langle 0, \dots, 0 \rangle$;
$\sigma \leftarrow \langle 0, \dots, 0 \rangle$;
$T \leftarrow \langle 2, \dots, n \rangle$;
$B \leftarrow \langle 2, \dots, n \rangle$;
$new\_win \leftarrow 0$;
$calculations \leftarrow 1$;
while $calculations > 0$ **do**
    $new\_win \leftarrow new\_win + win$;
    $calculations \leftarrow 0$;
    $b \leftarrow 1$;
    $new\_b \leftarrow 1$;
    for $j = 2$ **to** n **do**
        if $new\_b = B[j-1]$ **then**
            $B[j-1] \leftarrow b$;
            $b \leftarrow T[j-1]$;
            $new\_b \leftarrow T[j-1]$;
            continue;
        if $i - new\_b > new\_win$ **then**
            $B[j-1] \leftarrow j - new\_win$;
            $b \leftarrow j - new\_win$;
        else
            $B[j-1] \leftarrow b$;
            $b \leftarrow new\_b$;
        $new\_b \leftarrow T[j-1]$;
        for $i = T[j-1] - 1$ **downto** b **do**
            $w \leftarrow w(i, j)$;
            $\sigma[j-1] \leftarrow \sigma[j-1] + w$;
            $W[i] \leftarrow W[i] + \sigma[j-1]$;
            if $w = 1 - d$ **and** $W[i] \geq 0$ **then**
                $H \leftarrow H \cup \langle s_i, \dots, s_j \rangle$;
            if $\bar{w}_{max}(i, j) \geq 0$ **then**
                $new\_b \leftarrow i$;
            $calculations \leftarrow calculations + 1$;
        $T[j-1] \leftarrow b$;
return H;

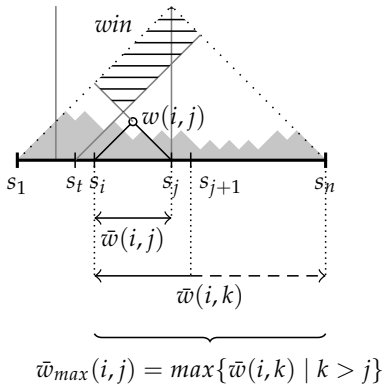


Figure 4.4: Processing a chromosome with the MIG^{++} algorithm.

by a number of SNPs equal to *win*. Therefore, the number of computed SNP-pair weights increases proportionally. This allows a more precise estimation of the upper bounds for the region weights with every new iteration. Figure 4.4 illustrates the pruning step in MIG^{++}. The SNP-pair weights computed at the previous stages are distributed uniformly across entire chromosome and are colored in gray. The SNP-pair weights within the hatched area are omitted from computation if $\bar{w}_{max}(i,j) < 0$ for any region that starts before the SNP s_t and ends after the SNP s_j, or s_t is out of the window *win* (in the latter case the omitted SNP-pair weights may be used in the next iteration).

By considering all SNP-pair weights computed in all previous iterations for the estimation of the upper bound, $\bar{w}_{max}(i,j)$, the algorithm requires linear time for each individual SNP pair to sum up all weights inside the corresponding region. We use a computationally cheaper constant-time solution, though it may lead to a less accurate estimation. Since $\bar{w}(i,k) \leq \bar{w}(i,j) + \bar{w}(1,k) - \bar{w}(1,j)$, we have $\max\{\bar{w}(i,k) \mid k > j\} \leq \bar{w}(i,j) + \max\{\bar{w}(1,k) \mid k > j\} - \bar{w}(1,j)$. An upper bound $\bar{w}_{max}(i,j)$ can then be computed as follows:

$$\bar{w}_{max}(i,j) = \bar{w}(i,j) + \max\{\bar{w}(1,k) \mid k > j\} - \bar{w}(1,j). \tag{4.2}$$

$\max\{\bar{w}(1,k) \mid k > j\}$ is computed in linear time after every scan of the chromosome, whereas $\bar{w}(1,j)$ is computed in constant time. Thus, the computation of the upper bound $\bar{w}_{max}(i,j)$ for each individual SNP pair requires only constant time.

When $win = n$, MIG^{++} is identical to MIG^{+}. When $win = 1$, the number of iterations becomes too large, introducing a significant computational burden. We propose to set $win = \lceil (n-1)(1-d)/2 \rceil$, that corresponds to $1-d$ percent of all SNP pairs, which is the minimal fraction of SNP pairs that must be considered before one can be sure that an n-SNP segment is not a haplotype block.

The MIG^{++} performs at most $\lambda n(n-1)/2$ computations, where λ, $1 - d \leq \lambda \leq 1$, depends on the data. However, the value of λ obtained with the MIG^{++} algorithm is expected to be always smaller than that from the MIG^{+} algorithm, because of the more precise estimation of $\bar{w}_{max}(i,j)$.

4.5 Experimental Evaluation

The experimental evaluation was based on the phased CEPH genotypes included in the HapMap phase II (HapMapII) [40] and the 1000 Genomes Project phase 1 release 3 (1000G) [24] databases. The HapMapII dataset included 2,543,857 SNPs from 120 haplotypes (60 individuals) and the 1000G dataset included 10,858,788 SNPs from 170 haplotypes (85 individuals).

To compare the new algorithms to the standard Haploview, in terms of runtime and memory usage, the ideal solution would have been that of randomly sampling regions with different characteristics from the HapMapII or 1000G datasets. However, the Haploview algorithm was so computationally expensive that it prohibited to consider a sufficiently large number of random regions and, therefore, to obtain a representative sample of all possible scenarios over the whole genome. For this reason, we selected the regions such that the most extreme scenarios, in terms of median SNP minor allele frequency (MAF) and median inter-SNP distance, were covered. To identify such representative regions, we performed the systematic scan of all SNPs in the genome using a sliding window of 1,000 SNPs, after removing chromosomal centromeres and the HLA region. For each sliding region, the median MAF and inter-SNP distance were recorded (Figure A.1). All regions were then represented in a two-dimensional Euclidean space, where the normalized inter-SNP distance was plotted against the normalized median MAF (Figure A.2). A total of nine regions were chosen for the experiments: the eight regions located on the outermost boundaries of the Euclidean space and the region closest to the center of the space. These regions represent scenarios with extreme and moderate median MAF and median inter-SNP distance. The procedure was repeated using larger sliding windows of 5,000 to 30,000 SNPs. If not stated otherwise, in the experimental results we report median values over the nine regions for every different window size.

The block partitions obtained with the WP and AV methods for $|D'_{i,j}|$ CI estimation were compared in terms of total number of blocks, median number of SNPs per block, proportion of SNPs clustered into blocks, and median within block haplotype diversity. Haplotype diversity [1, 27] is defined as the ratio between the number of common haplotypes and the total number of haplotypes within a block. Common haplotypes are those occurring more than once. The haplotype diversity index ranges from 0

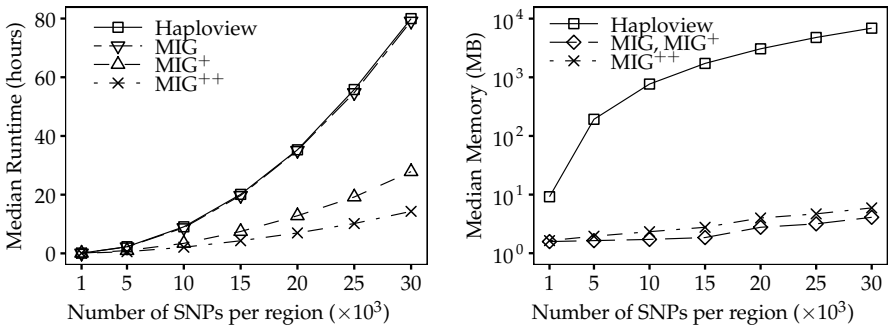


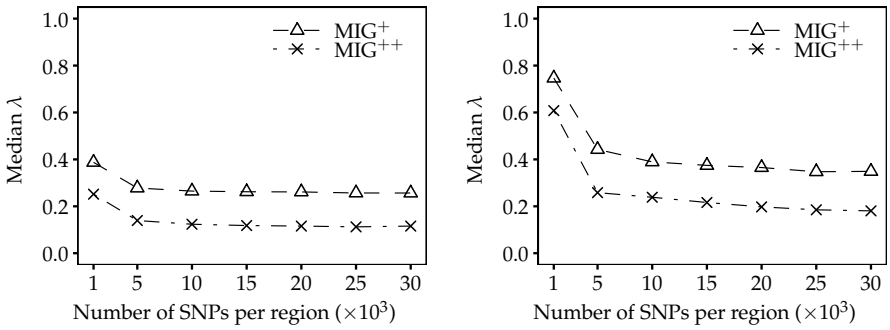
Figure 4.5: Performance of the algorithms with the WP method, when applied to the 1000G dataset.

(complete diversity) to 1 (no diversity).

The three MIG algorithms were implemented in C++. To guarantee a fair comparison, the original Java implementation of the Haploview algorithm was rewritten in C++, too. By default, Haploview considers only SNP pairs within a maximal distance of 500 Kbp. We removed this constraint because it could affect the block partitions of very wide regions. The confidence intervals of $|D'_{i,j}|$ were estimated using the Wall and Pritchard method [20] based on a likelihood function evaluated on a 1,000 points grid (100 in the original Haploview implementation). We didn't consider the population specific two-, three-, and four-marker rules, proposed by Gabriel *et al.* [13] when very short regions are processed, because they have no impact on the computational efficiency of the algorithms. All experiments were run on a machine with an Opteron 8356 Quad Core (2.3GHz) CPU.

4.5.1 Runtime and Memory Usage with the WP Method

Figure 4.5 shows runtime and memory performance of Haploview and the three MIG algorithms based on the WP method, when applied to the 1000G dataset. Since both Haploview and MIG perform $n(n-1)/2$ computations, it was expected to see identical runtime: both of them took 80 hours to process regions of 30,000 SNPs. However, MIG used three orders of magnitude less memory than Haploview (3 MB vs. 7 GB). The runtime was significantly reduced with MIG$^+$ (27 hours) and even further with MIG^{++} (14 hours). The runtime difference between algorithms increased with the region size (number of SNPs). Memory usage was identical for MIG and MIG$^+$, whereas MIG^{++} required slightly more memory to store the computational status between iterative region scans. Similar results were obtained on the HapMapII dataset (results not shown).



(a) HapMapII (b) 1000G

Figure 4.6: The λ pruning coefficient for MIG^{+} and MIG^{++} with the WP method.

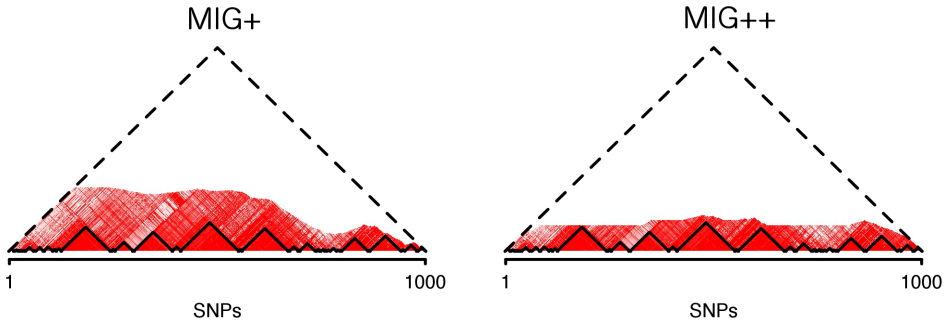
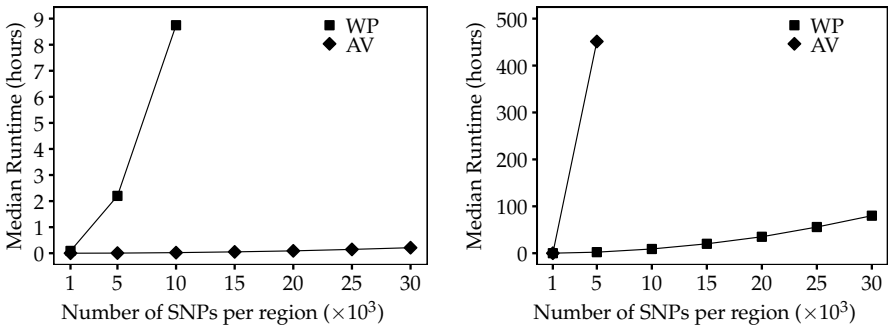


Figure 4.7: LD heatmap of chr20:31,767,872-33,700,401 in the HapMapII dataset, which contains 1,000 polymorphic SNPs.

The MIG^{++} omitted more unnecessary computations than MIG^{+}, which is reflected by the smaller λ coefficient in both HapMapII and 1000G datasets (Figure 4.6). The λ values decreased with increasing number of SNPs in the region. When increasing the region size, after a rapid decline for small regions, λ reached stable values with both MIG^{+} and MIG^{++} algorithms and in both datasets. This behavior relates to the LD decay with distance. In regions of 30,000 SNPs in the 1000G dataset, the MIG^{++} algorithm was able to omit $\sim$80% of the calculations ($\lambda\sim$0.20), while MIG^{+} could omit $\sim$60% of the calculations ($\lambda\sim$0.40). An example of the reduction of the number of calculations is given in Figure 4.7, where MIG^{+} and MIG^{++} are compared to Haploview, which is represented by the entire triangle.



(a) MIG (b) Haploview

Figure 4.8: Impact of the WP and AV methods on runtime, when applied to the 1000G dataset.

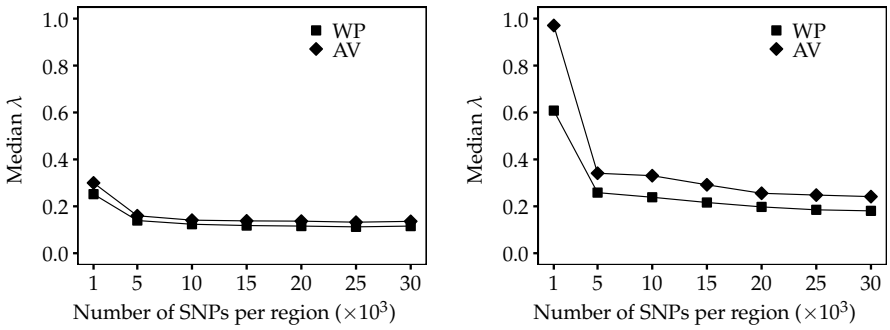
4.5.2 Runtime and Memory Usage with the AV Method

When we introduced the AV method to estimate the $|D'_{i,j}|$ CI, we observed a drastic reduction of the computational time of the MIG algorithm. With the AV approach, the median runtime needed to analyze sequences of 10,000 SNPs in the 1000G dataset was of 2 minutes. The same analysis took a median of 8.7 hours with the WP method (Figure 4.8(a)). Proportional time reduction was observed for MIG$^+$ and MIG^{++}. Similar results were obtained in the HapMapII dataset (results not shown).

We observed that the introduction of the AV method caused a slight increase of the λ coefficient (Figure 4.9). This is because, with the AV method, more SNP pairs are classified to be in *strong LD*. This causes an increase of the number of possible configurations to be checked and results in a larger set of candidate haplotype blocks. With the AV method, the MIG algorithms identified tens of millions of candidate haplotype blocks (Figure 4.10). The number of candidate blocks was even larger when the AV method was applied directly to Haploview, where candidate blocks need to satisfy only Definition 2(a). This significantly larger number of candidate blocks explains the increase in runtime of Haploview when using the AV method: for regions of 5,000 SNPs in the 1000G dataset, the median runtime was of 451 hours with the AV against the 2 hours with the WP method (Figure 4.8(b)).

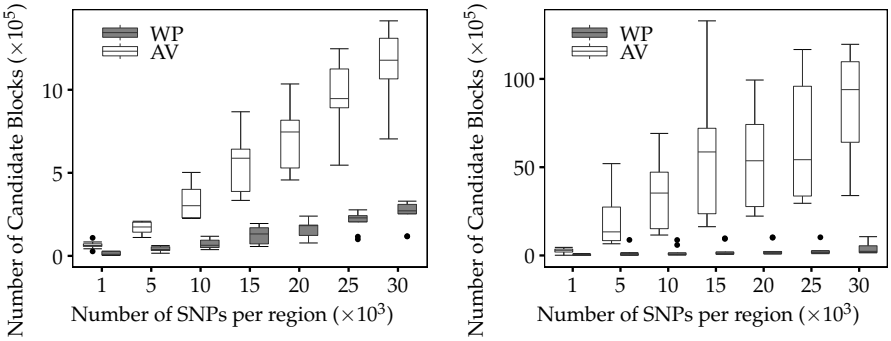
4.5.3 Block Partitions with the WP and AV Methods

The characteristics of the different block partitions obtained with the WP and AV methods are summarized in Figure 4.11. The AV method produced a smaller number of blocks than the WP method (top panels). The median number of blocks per region increased along with the number of SNPs,



(a) HapMapII (b) 1000G

Figure 4.9: The λ pruning coefficient for MIG^{++}: comparison between WP and AV methods.

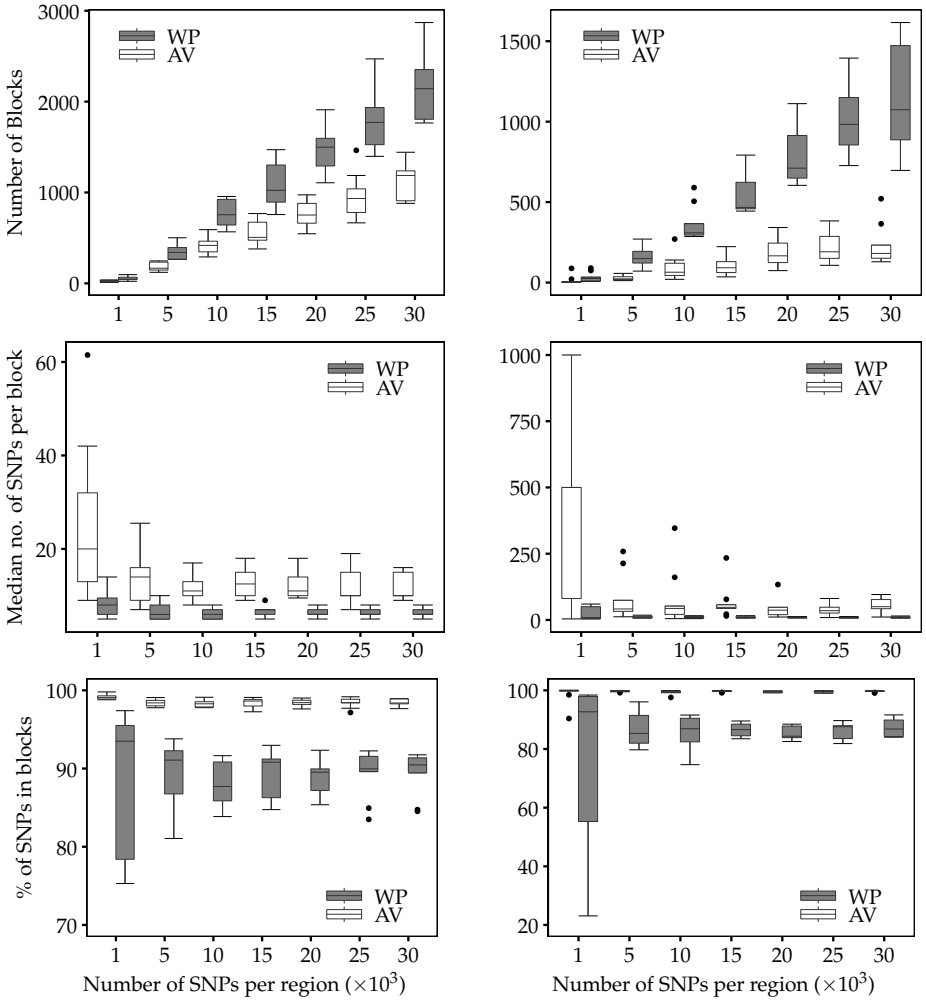


(a) HapMapII (b) 1000G

Figure 4.10: Number of candidate haplotype blocks detected by the MIG, MIG^{+} and MIG^{++} algorithms with the WP and AV methods.

and it increased faster for the WP compared to the AV method. Considering the median number of SNPs per block, the AV method produced larger blocks than the WP method (middle panels). For very short regions (e.g., 1,000 SNPs) both methods generally induced larger blocks. This is because such small regions might be completely covered by a single or very few haplotype blocks. The median number of SNPs per block decreased along with the increase of the length of the region considered. Overall, the AV method assigned a higher percentage of SNPs to blocks compared to the WP method, which left more singleton SNPs outside of any block (bottom panels). In the analysis of the HapMapII dataset, 98.4% of the SNPs were clustered within blocks with the AV method and 90.5% with the WP method. In the analysis of the 1000G dataset, the percentages were of 99.7 and 86.8, respectively.

We observed that 100% of the blocks identified by the WP method were

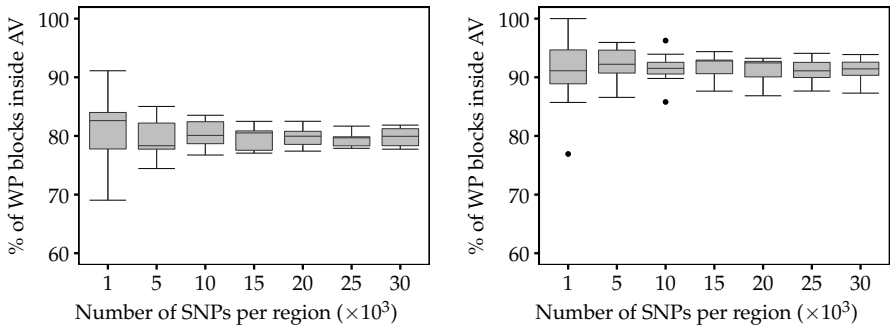


(a) HapMapII (b) 1000G

Figure 4.11: Haplotype block characteristics of WP and AV methods.

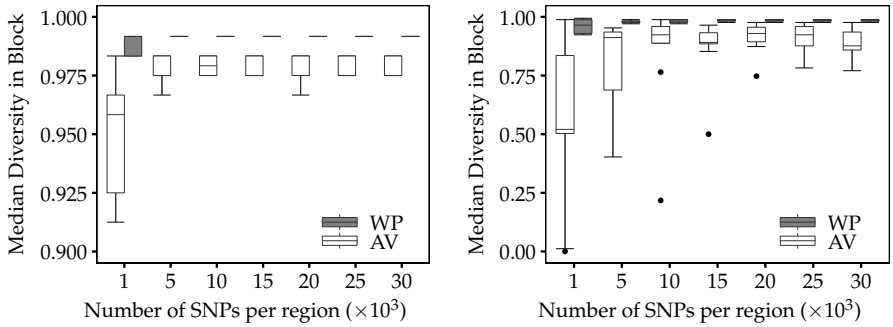
overlapping blocks identified by the AV method. More specifically, 80% to 90% of the blocks based on the WP method were completely included within blocks based on the AV method (Figure 4.12). The remaining 10% to 20% of WP blocks whose borders were crossing borders of AV blocks, could be entirely attributed to the selection mechanism in the step (2) of the algorithm, when larger candidate blocks are prioritized over the shorter ones. In fact, when, instead of looking at the final block partition, we focused on the intermediate set of candidate blocks before the final pruning, we observed that 100% of the candidates from the WP method were entirely included within the candidate blocks from the AV method.

Consistently with the findings of larger AV blocks, we observed a gen-



(a) HapMapII (b) 1000G

Figure 4.12: Number of blocks detected with the WP method that are completely inside blocks detected with the AV method.



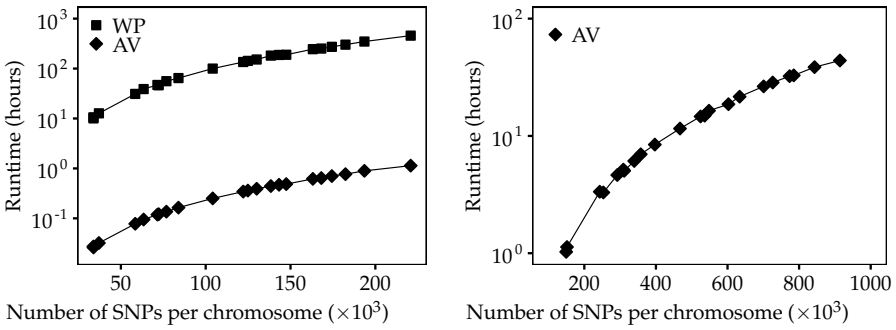
(a) HapMapII (b) 1000G

Figure 4.13: Within-block haplotype diversity with WP and AV methods.

erally higher haplotype diversity in the partitions obtained with the AV method compared to the WP method (Figure 4.13). For instance, when considering regions of 30,000 SNPs in the 1000G dataset, we observed median within-block haplotype diversity indices of 0.876 and 0.982 with the AV and WP methods, respectively. Slightly higher diversity indices were observed in the HapMapII dataset: 0.975 and 0.992 for the AV and WP methods, respectively. The within-block diversity was more variable in short than in long regions because, as observed above, when regions are too small, then it might be difficult to identify more than one block.

4.5.4 Whole Genome Partition

The linear memory complexity and the significant reduction of the number of computations allowed us to run MIG^{++} on a genome-wide scale. We could run MIG^{++} on the full HapMapII dataset using both the WP and AV methods for $|D'_{i,j}|$ CI derivation. Using the more efficient AV method,



(a) HapMapII (b) 1000G

Figure 4.14: Runtime of the MIG^{++} algorithm on whole-genome data.

we were also able to run a genome-wide haplotype block partition of the complete 1000G dataset. The runtime for the two datasets is shown in Figure 4.14. For HapMapII, the maximal runtime was of 1 hour when using the AV method and of 457 hours when using the WP method. In both cases, the maximal runtime was observed for chromosome 2, which contained 220,833 SNPs. The median λ value across all chromosomes was 0.129 (min=0.125, max=0.133) for the AV method and 0.103 (min=0.099, max=0.110) for the WP method. For the 1000G dataset, the maximal runtime using the AV method was of 44 hours on chromosome 2, which contained 913,923 SNPs. The median λ value across all chromosomes was 0.216 (min=0.206, max=0.224). The maximal memory usage was very low and didn't exceed 151 MB and 3.6 GB for the HapMapII and 1000G datasets, respectively.

Figure 4.15 shows the number of haplotype blocks per chromosome. In the HapMapII dataset, the largest number of blocks occurred in chromosome 2: 14,164 blocks with the WP method and 7,482 blocks with the AV method. The number of blocks detected with the WP method was always exceeding the number of blocks detected with the AV method. For some chromosomes, the partitions obtained by the WP method contained almost twice as many blocks as the partitions obtained by the AV method. When using the AV method, we detected a very similar number of blocks in the HapMapII and 1000G datasets. Across all chromosomes, 100% of the blocks detected with the WP method were overlapping blocks detected with the AV method. The median percentage of the WP blocks completely covered by the AV blocks was 0.797 (min=0.773, max=0.813).

The characteristics of the whole-genome block partitions obtained with the AV and WP methods are summarized in Table 4.1. The results were similar to the experiments on smaller regions. In the HapMapII dataset, fewer and larger blocks were detected with the AV method than with the WP method. With the AV method, a higher percentage of SNPs was assigned to

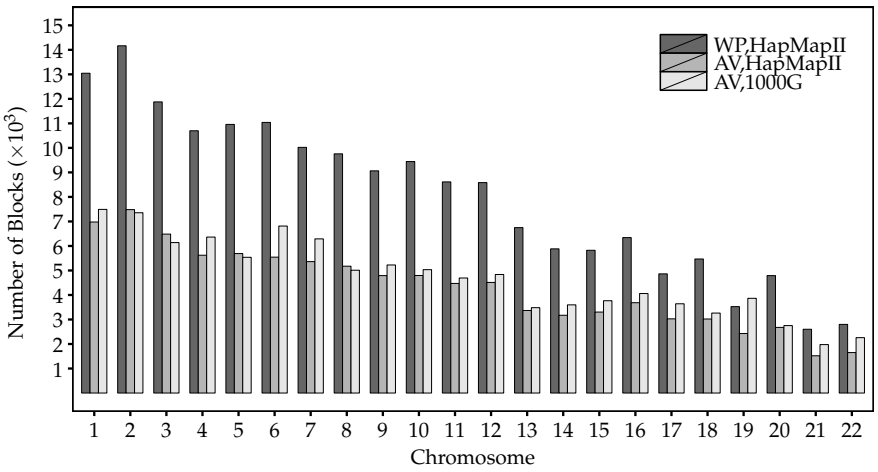


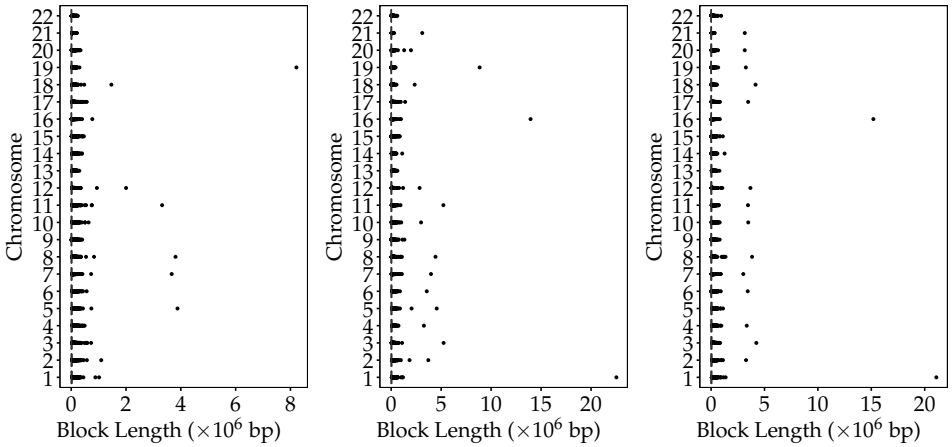
Figure 4.15: Number of haplotype blocks in the HapMapII and 1000G datasets when the D' CIs are estimated with the WP and AV methods.

	WP			AV		
	median	min	max	median	min	max
			HapMapII			
No. of SNPs per Block	6	5	7	12	7	14
% of SNPs in Blocks	89.6	85.8	91.4	98.5	97.2	98.8
Diversity in Block	0.992	0.992	0.992	0.975	0.975	0.983
			1000G			
No. of SNPs per Block	–	–	–	25	13	34
% of SNPs in Blocks	–	–	–	99.4	99.1	99.6
Diversity in Block	–	–	–	0.944	0.929	0.971

Table 4.1: Characteristics of the whole-genome haplotype block partitions obtained with the WP and AV methods.

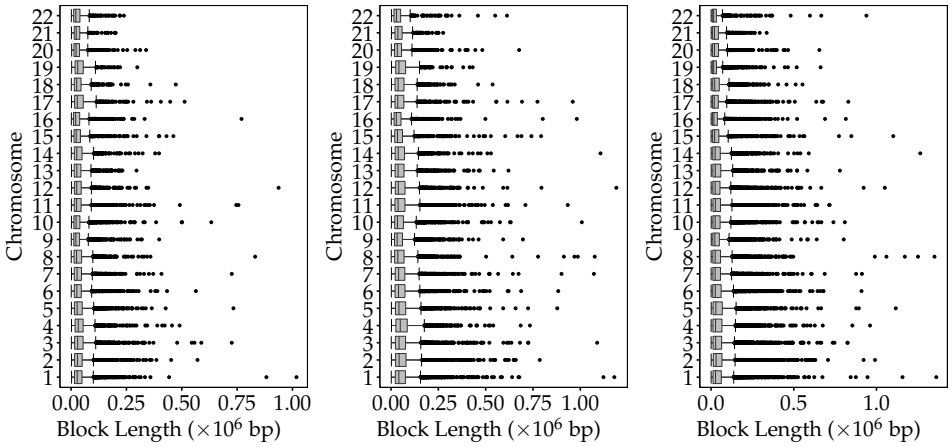
blocks and the within-block haplotype diversity index was slightly smaller. However, the haplotype diversity was close to one for both methods, indicating that in both cases the number of possible haplotypes should be very limited. When applying the AV-based MIG^{++} algorithm to the 1000G dataset, we observed a higher percentage of SNPs in blocks and a slightly smaller diversity index, which is explained by the higher number of SNPs per block.

For both HapMapII and 1000G datasets, the largest blocks were located over the chromosomal centromeres and spanned tens of millions of base-pairs (bp) (Figure 4.16). Some of these very large blocks were characterized by very low and irregular SNP density. After filtering out these exceptionally large blocks, the largest block identified by the WP-based MIG^{++} algorithm in the HapMapII dataset was located in chromosome 1, it was 1,017,844 bp long and included 398 SNPs (Figure 4.17). When using the AV method, the largest block was located in chromosome 12, it was



(a) WP, HapMapII (b) AV, HapMapII (c) AV, 1000G

Figure 4.16: Lengths of the blocks in base pairs (bp) estimated with the WP and AV methods on the complete HapMapII and 1000G datasets.



(a) WP, HapMapII (b) AV, HapMapII (c) AV, 1000G

Figure 4.17: Lengths of the blocks in base pairs (bp) estimated with the WP and AV methods on the complete HapMapII and 1000G datasets that do not overlap centromeres and have a maximal distance between nearby SNPs of at most 1/5 of the total block length.

1,190,412 bp long and included 335 SNPs. In the 1000G dataset, the largest block detected by the AV-based MIG^{++} was located in chromosome 1, it was 1,361,781 bp long and included 2,896 SNPs.

4.6 Summary

We propose an algorithm for haplotype block partitioning, termed MIG^{++}, which represents a scalable implementation of the Haploview algorithm and produces the same results in a much shorter time and using a substantially smaller amount of main memory. MIG^{++} can process large DNA regions using only a handful of megabytes of main memory. In such situations, Haploview would require gigabytes. In terms of runtime, the MIG^{++} is several times faster than Haploview. We also demonstrated that more than 80% of calculations were not necessary for the purpose of block recognition and could be omitted, thus achieving a higher efficiency. The improved performance of the algorithm makes it possible to process very large chromosomal segments. When the approximated variance estimator, proposed by Zapata *et al.* [21], is used to estimate the $|D'_{i,j}|$ CI, the MIG^{++} can be applied genome-wide and process high density datasets, such as the 1000G, in a very short time.

With its very small memory requirements, the MIG^{++} can process any number of SNPs. This allowed us to avoid Haploview's restrictions on the maximal haplotype block length (the default limitation is 500Kbp) and to consider the LD between SNPs at any distance. Our whole-genome experiments showed that the haplotype blocks, based on the Gabriel *et al.* [13] definition, can span more than 500Kbp and can extend over several millions of base pairs. This empirical result suggests that limiting the maximal block length may alter the block partition. The alteration can be substantial because the algorithm prioritizes the largest blocks. The smallest blocks are retained only when they do not overlap with the largest ones. For this reason, to constrain the block length within pre-specified limits may induce a cascade of effects and may affect the final partition of very large segments. This is relevant, for example, when assessing the LD pattern of loci selected from GWAS, with the aim of identifying genes related to the lead SNP. In such cases, different partitions could imply different genes to be selected for follow-up.

With the MIG^{++} algorithm, we were able to run a haplotype block recognition of the entire HapMapII dataset. However, it still required an unacceptably long time to apply the algorithm to larger and denser genomes, such as the 1000G dataset. This limitation is due to the use of the Wall and Pritchard [20] method, which models the $|D'_{i,j}|$ likelihood and derives the $|D'_{i,j}|$ CIs using an iterative procedure. In contrast, if the $D'_{i,j}$ variance is estimated with the approximated formula suggested by Zapata *et al.* [21], it is possible to derive the $|D'_{i,j}|$ CI with a single mathematical calculation. Thanks to this computationally less demanding solution, we could perform a complete block recognition of the HapMapII dataset in 1 hour and to process the entire 1000G dataset in 44 hours. To the best of

our knowledge, this is the first time that such a marker-dense genome has been partitioned with a threshold-free approach. Previously, block partition of the whole genome could only be achieved by dividing chromosomes into small chunks or by restricting computations using sliding window approaches. Such choices may introduce artificial breaks to the real haplotype structure.

It is important to note that the block partition obtained with an algorithm based on the AV method is not fully equivalent to a partition obtained based on the WP method. For large sample sizes and for common variants, the estimated variance of the $D'_{i,j}$ statistic is going to be similar, whichever method is used. However, when crossing a common with a rare SNP, it often happens that one of the four possible haplotypes is not present in the sample. In such situations, it is very likely that the $|D'_{i,j}|$ CI shrinks to 1 because the approximated variance is zero. In this way, the SNP pair is systematically classified as a *strong LD* pair. As a result, SNPs with rare alleles are easily grouped together into very large blocks, boosting the region coverage and the median number of SNPs per block. The WP method is less sensitive to extreme $D'_{i,j}$ values, and the resulting blocks are generally shorter. However, we observed that most (80%) of the haplotype blocks obtained with the WP method were contained within the larger blocks obtained with the AV method. That is, the use of the AV method produced a coarser partition, where AV blocks entirely contained one or more WP blocks. For this reason, the AV blocks showed a higher haplotype diversity, in the terms described by Patil *et al.* [1] and Zhang *et al.* [27], than the WP blocks.

The MIG algorithms are available in the *LDExplorer* R package at The Comprehensive R Archive Network (CRAN) [41] together with usage instructions and examples. Also, they were adopted by the recent version 1.9.0 of PLINK [53], which is one of the most widely used software applications for genetic association studies worldwide.

Chapter 5

Sampling-based Computation of Haplotype Blocks

Our general aim was to further improve performance of the MIG^{++} algorithm in order to avoid the need to use an approximation of the $D'_{i,j}$ variance. Actually, the approximation method by Zapata [21] works very well asymptotically, but introduces substantial departures from the original likelihood-based approach implemented in Haploview [14] for extreme $D'_{i,j}$ values, such as those observed between common and rare SNPs. Since such situations are going to become common due to the spread of data from high-throughput sequencing technologies, the difference in estimations of the $D'_{i,j}$ variance is going to produce largely different block partitions. Thus, a runtime efficient algorithm that avoids the use of approximated $D'_{i,j}$ variance estimators in favor of the original likelihood-based method would be desirable. In this chapter, we introduce a new algorithm, termed *S-MIG*$^{++}$, that significantly improves all previously proposed solutions in terms of runtime.

5.1 Overview

The basic idea of our novel algorithm is the following. Imagine that we want to verify that a given set of regions consists only of true haplotype blocks. In order to do this, it is sufficient to compute the LD coefficients only for those SNP pairs that are covered by these regions. Thus, a significant part of irrelevant computations is omitted. Since the set of true haplotype blocks is unknown, we use sampling to estimate a set of regions that with a high probability include the true haplotype blocks. Then, only the SNP pairs inside the estimated regions need to be considered to obtain the true haplotype blocks. Hence, the algorithm works in two steps:

- Firstly, by sampling only a tiny fraction of all SNP pairs in a chromo-

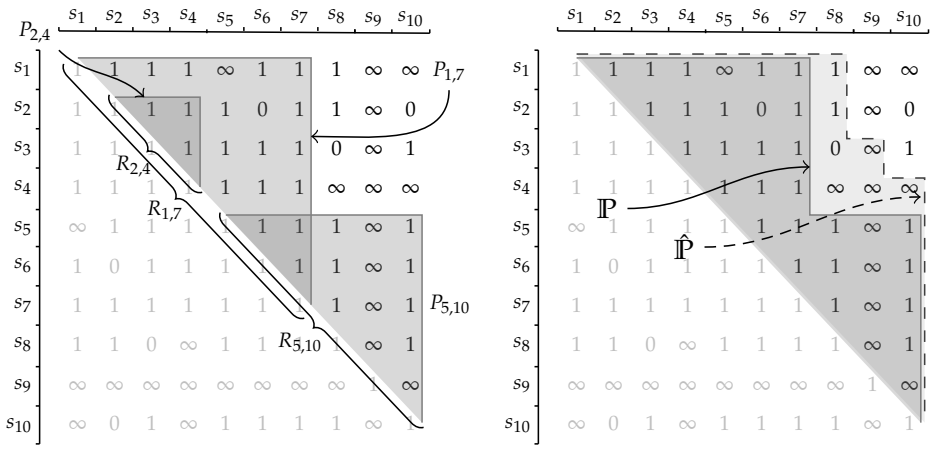


Figure 5.1: The LD matrix, $\mathsf{T}_{10\times10}$, of a chromosome $S = \langle s_1, \ldots, s_{10}\rangle$ consisting of ten SNPs.

some, correct upper limits of the haplotype block boundaries, which we call *haplotype block contour*, are estimated with a very high probability.

- Secondly, within the estimated upper limits, the exact boundaries of the haplotype blocks are computed; we refer to this step as *haplotype blocks refinement*.

We represent the classified SNP pairs of a chromosome in an LD matrix.

Definition 5. (*LD Matrix*). Let $S = \langle s_1, \ldots, s_n\rangle$ be a chromosome. The *LD matrix*, $\mathsf{T}_{n\times n}$, of S is defined as

$$t_{i,j} = \begin{cases} 1 & \text{if } s_i \text{ and } s_j \text{ is a } \textit{strong LD} \text{ pair,} \\ 0 & \text{if } s_i \text{ and } s_j \text{ is a } \textit{strong EHR} \text{ pair,} \\ \infty & \text{if } s_i \text{ and } s_j \text{ is a } \textit{non-informative} \text{ pair.} \end{cases}$$

Since the LD matrix is symmetric, we consider only the upper triangle of the matrix, excluding the diagonal. An example is given in Figure 5.1.

To represent the set of all SNP pairs within a particular chromosomal region, we introduce the concept of region profile.

Definition 6. (*Region Profile*). Let $S = \langle s_1, \ldots, s_n\rangle$ be a chromosome and $R_{i,j} = \langle s_i, \ldots, s_j\rangle$ be a region in S. The *region profile* of $R_{i,j}$ is defined as the set $P_{i,j} = \{(s_{i'}, s_{j'}) \mid i \leq i' < j' \leq j\}$.

The LD matrix elements that correspond to the SNP pairs in a region profile form a triangle whose base lies on the matrix diagonal.

Example 2. The gray areas in Figure 5.1(a) illustrate the profiles of three regions (out of a total of 45 regions for a chromosome of length ten). For instance, the region $R_{2,4} = \langle s_2, \ldots, s_4 \rangle$ is a subsequence of three consecutive SNPs. The corresponding profile is $P_{2,4} = \{(s_2, s_3), (s_2, s_4), (s_3, s_4)\}$.

Definition 7. (*LD/I Ratio*). Let $\mathsf{T}_{n \times n}$ be the LD matrix of a chromosome $S = \langle s_1, \ldots, s_n \rangle$ and $R_{i,j}$ be a region with profile $P_{i,j}$. We define the ratio of *strong LD* to *informative* SNP pairs (*LD/I* ratio) as

$$\rho_{i,j} = \frac{|\{(s_{i'}, s_{j'}) \in P_{i,j} \mid t_{i',j'} = 1\}|}{|\{(s_{i'}, s_{j'}) \in P_{i,j} \mid t_{i',j'} \in \{0,1\}\}|}.$$

Thus, $\rho_{i,j}$ is the proportion of *strong LD* SNP pairs in the region $R_{i,j}$ with respect only to the *informative* SNP pairs in this region. Using the above concepts, Definition 2 of a haplotype block can be reformulated as follows.

Definition 8. (*Haplotype Block*). A region $R_{i,j} = \langle s_i, \ldots, s_j \rangle$ in a chromosome $S = \langle s_1, \ldots, s_n \rangle$ is a *haplotype block* if

(a) $t_{i,j} = 1$ and

(b) $\rho_{i,j} \geq 0.95$.

Example 3. The $R_{1,7}$ region in Figure 5.1(a) is a haplotype block since $t_{1,7} = 1$ and $\rho_{1,7} = 19/20 = 0.95$. Similarly, regions $R_{5,10}$ and $R_{2,4}$ are haplotype blocks. $R_{1,7}$ and $R_{5,10}$ are overlapping, whereas $R_{2,4}$ is completely contained in $R_{1,7}$. An example of a region that is contained in a haplotype block but does not form itself a haplotype block is $R_{2,6}$, since $t_{2,6} = 0$. In total, the LD matrix in Figure 5.1(a) contains 24 haplotype blocks.

Imagine that the true haplotype blocks in a chromosome would be known in advance. Then, we can define the set of SNP pairs that are inside all region profiles corresponding to these haplotype blocks. We call this set the haplotype block contour.

Definition 9. (*Haplotype Block Contour*). Let $R_{i_1,j_1}, \ldots, R_{i_k,j_k}$ be the list of all haplotype blocks in a chromosome S with corresponding profiles $P_{i_1,j_1}, \ldots, P_{i_k,j_k}$. The set $\mathbb{P} = P_{i_1,j_1} \cup \cdots \cup P_{i_k,j_k}$ is termed the *haplotype block contour* of chromosome S.

Example 4. In Figure 5.1(b), the regions $R_{1,7}$ and $R_{5,10}$ are the two longest haplotype blocks, and together they completely cover the remaining 22 haplotype blocks. Hence, the haplotype block contour is $\mathbb{P} = P_{1,7} \cup P_{5,10} = \{(s_1, s_2), (s_1, s_3), \ldots, (s_9, s_{10})\}$, containing a total of 33 SNP pairs.

From Definition 9 follows that no SNP pair outside the contour can form a haplotype block. Therefore, an ideal haplotype blocks recognition algorithm would compute at most $|\mathbb{P}|$ LD coefficients, i.e., for all SNP pairs in

$\mathbb{P}$. The Haploview algorithm always computes all $n \cdot (n-1)/2$ LD coefficients, i.e., the entire LD matrix. The MIG^{++} algorithm computes significantly less, though still many more than an ideal solution would. In general, the runtime efficiency depends on how close the number of actually computed LD coefficients is to $|\mathbb{P}|$. Based on this observation, we propose an improved algorithm that has two steps:

1. *haplotype block contour estimation*,
2. *haplotype blocks refinement*.

The first step computes an estimation of the true haplotype block contour, that is a set of SNP pairs that with a very high probability completely cover the true haplotype block contour. In Figure 5.1(b), the dashed line outlines the estimated haplotype block contour, $\hat{\mathbb{P}}$. It completely covers the true contour $\mathbb{P}$, but contains seven additional SNP pairs. During the haplotype blocks refinement in the second step, the estimated contour is used as an effective way to prune the search space. That is, for the computation of the exact haplotype blocks, only SNP pairs in the estimated contour $\hat{\mathbb{P}}$ are considered. Thus, the number of LD computations is upper bounded by the cardinality of the estimated contour, $|\hat{\mathbb{P}}|$, which yields significant improvements with respect to MIG^{++}.

The following sections describe the two steps in detail. The pseudocode of the new algorithm, S-MIG^{++}, is given in Algorithm 4.

5.2 Haplotype Block Contour Estimation

5.2.1 Overview

The computation of the estimated haplotype block contour, $\hat{\mathbb{P}}$, works in three steps:

1. *Chromosome splitting.* A chromosome is split into k segments, inducing a grid of $k \times k$ cells in the LD matrix $\mathsf{T}_{n \times n}$.
2. *SNP pairs sampling.* In each cell, a fraction $\sigma \in (0, 1)$ of SNP pairs is sampled and their LD coefficients are computed. The number of sampled *strong LD*, *strong EHR* and *non-informative* pairs is counted for every cell.
3. *Haplotype block contour estimation.* Based on the sampled data, the *LD/I* ratio is estimated for those regions that are composed of a whole number of consecutive segments. If the estimated *LD/I* ratio satisfies Definition 8(b) of a haplotype block, the SNP pairs in the corresponding profile are added to the estimated contour.

Algorithm 4: S-MIG^{++}

Data: $S = \langle s_1, \dots s_n \rangle$
Input: p, k, σ
Result: $H = \emptyset$

$l \leftarrow n/k$;
$\alpha \leftarrow 1 - p^{2/(k(k+1))}$;
$\hat{\mathbb{P}} \leftarrow \emptyset$;

/* Step 1: Haplotype block contour estimation. */
/* Sub-step 1.1: Chromosome splitting. $\mathsf{T}_{n \times n}$ is never stored, we store only cell dimensions. */
for $x = 1$ **to** k **do**
  for $y = x$ **to** k **do**
    $rows \leftarrow \{(x-1) \cdot l + 1, \dots, x \cdot l\}$;
    $columns \leftarrow \{(y-1) \cdot l + 1, \dots, y \cdot l\}$;
    $C_{x,y} \leftarrow \mathsf{T}_{n \times n}[rows, columns]$;

/* Sub-step 1.2: SNP pairs sampling. */
for $x = 1$ **to** k **do**
  for $y = x$ **to** k **do**
    $m_{x,y} \leftarrow sample(cell = C_{x,y}, samples = \sigma \cdot sizeof(C_{x,y}))$;

/* Sub-step 1.3: Haplotype block contour estimation. We traverse sampling matrix along its diagonal, so we are sure that for any coordinate (i, j) previous results at coordinate $(i + l, j - l)$ were already computed and can be aggregated. */
for $z = 0$ **to** $k - 1$ **do**
  for $x = z + 1$ **to** k **do**
    $y \leftarrow x - z$;
    $i \leftarrow (x-1) \cdot l + 1$;
    $j \leftarrow y \cdot l$;
    $\mathbf{m}_{i,j} \leftarrow (\mathbf{m}_1 = 0, \mathbf{m}_2 = 0, \mathbf{m}_3 = 0)$;
    for $x' = x$ **to** y **do**
      for $y' = x'$ **to** y **do**
        $\mathbf{m}_{i,j} \leftarrow \mathbf{m}_{i,j} + m_{x',y'}$;
    $m \leftarrow \mathbf{m}_{i,j}[1] + \mathbf{m}_{i,j}[2] + \mathbf{m}_{i,j}[3]$;
    $ub^{(1)}_{i,j} \leftarrow compute\_ci(strongLD = \mathbf{m}_{i,j}[1], samples = m, typeI = \alpha)$;
    $lb^{(2)}_{i,j} \leftarrow compute\_ci(strongEHR = \mathbf{m}_{i,j}[2], samples = m, typeI = \alpha)$;
    $\hat{\rho}_{i,j} \leftarrow estimate\_LD\_I\_ratio(ub^{(1)}_{i,j}, lb^{(2)}_{i+l,j-l}, |P_{i,j}|, P_{i+l,j-l})$;
    if $\hat{\rho}_{i,j} \geq 0.95$ **then**
      $\hat{\mathbb{P}} \leftarrow \hat{\mathbb{P}} \cup P_{i,j}$;

/* Step 2: Refinement. Pass $\hat{\mathbb{P}}$ to MIG^{++} and restrict its search space to elements from $\hat{\mathbb{P}}$. */
$H \leftarrow$ S-MIG$^{++}(\hat{\mathbb{P}})$

return H;

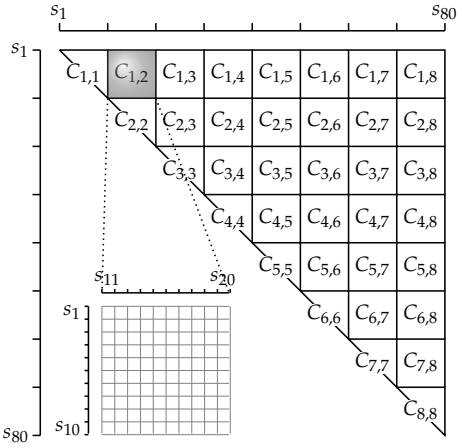


Figure 5.2: The uniform 8×8 grid of cells in $\mathsf{T}_{80\times80}$ LD matrix after the chromosome was split into 8 segments of 10 SNPs each.

The basic idea of this approach is (1) to reduce the dimensionality from n SNPs to k segments, thereby approximating the contour at the granularity of segments rather than SNPs, and (2) to use an efficient sampling-based method to compute an estimated haplotype block contour that completely covers the true contour; we refer to this also as *correctly estimated haplotype block contour*. We will show in the following how to guarantee that the probability of a correctly estimated contour is so high to give, practically, always identical results to the true contour.

The number of segments, k, and the sampling fraction, σ, control the precision of the estimated contour $\hat{\mathbb{P}}$. A larger k implies a finer granularity and, therefore, smoother borders of $\hat{\mathbb{P}}$. A larger σ results in more accurate LD information being captured and so in more precise estimates of the contour, with the drawback of increasing the runtime. The effect of different values of k and σ on runtime and precision of the estimated contour are analyzed experimentally in Section 5.4.

5.2.2 Chromosome Splitting

Without loss of generality, we assume that a chromosome is split into k equal-sized *segments*, each containing $l = \frac{n}{k}$ SNPs. In doing so, a grid of $k \times k$ *cells* is imposed on the LD matrix. Let's denote a cell by $C_{x,y}$, where $1 \leq x \leq y \leq k$. Each cell represents a sub-matrix of the LD matrix. Since we consider only the upper triangle of the matrix, there is a total of $k(k+1)/2$ cells. Each cell contains l^2 elements, except the diagonal cells, which contain $l(l+1)/2$ elements each.

Example 5. Figure 5.2 shows the split of the chromosome $S = \langle s_1, \ldots, s_{80} \rangle$ into $k = 8$ consecutive segments,

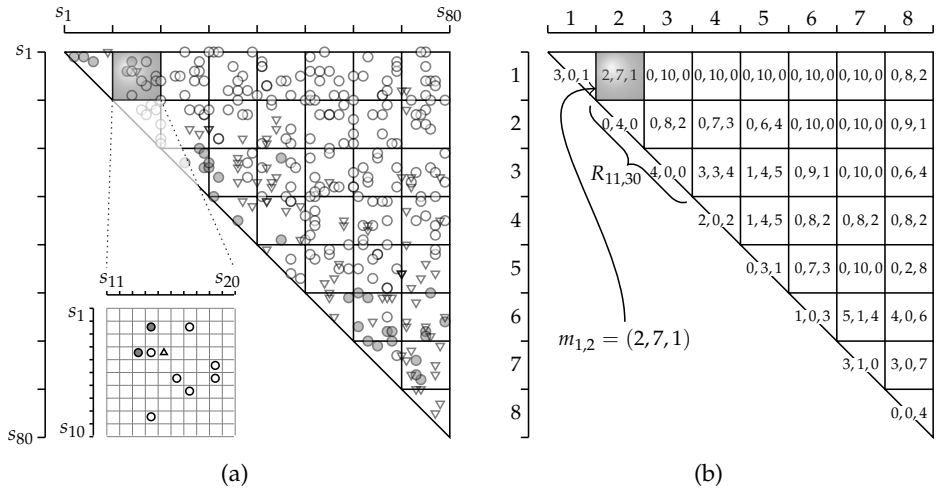


Figure 5.3: Sampling SNP pairs.

$\langle s_1, \ldots, s_{10} \rangle, \langle s_{11}, \ldots, s_{20} \rangle, \ldots, \langle s_{71}, \ldots, s_{80} \rangle$, each consisting of $l = 10$ SNPs. This segmentation in turn splits the LD matrix into a uniform grid of 36 cells. Each non-diagonal cell $C_{x,y}$, $x < y$, represents 100 SNP pairs, whereas each diagonal cell, $C_{x,x}$, represents 45 SNP pairs.

5.2.3 Sampling SNP Pairs

In each cell, a fraction σ of SNP pairs, where $0 < \sigma < 1$, is randomly sampled and the LD coefficients between these pairs are computed. The results of the sampling over each cell are stored in the so-called sampling matrix.

Definition 10. (*Sampling Matrix*). Let $\mathsf{T}_{n \times n}$ be an LD matrix that is split into a uniform grid of $k \times k$, $1 < k < n$, cells. The results of the sampling step are stored in a *sampling matrix*, $\mathsf{M}_{k \times k}$. Each entry stores a vector $m_{x,y} = (m_1, m_2, m_3)$, where m_1, m_2, and m_3 are, respectively, the numbers of sampled *strong LD*, *strong EHR*, and *non-informative* SNP pairs.

Example 6. Figure 5.3(a) shows the sampling of SNP pairs in the LD matrix. In each cell, the LD coefficients of 1% ($\sigma = 0.01$) of all SNP pairs are computed and classified (i.e., ten SNP pairs in each non-diagonal cell and four SNP pairs in each diagonal cell). For example, in cell $C_{1,2}$, two of the sampled SNP pairs are *strong LD* (dark circles), seven are *strong EHR* (white circles), and one is non-informative (triangle). The numbers of the sampled SNP pairs are stored in the sampling matrix in Figure 5.3(b).

5.2.4 Estimating the Haplotype Block Contour

Given that the haplotype block contour is defined as the union of all haplotype block profiles, if we follow a naïve approach to compute the estimated contour $\hat{\mathbb{P}}$, we would need to estimate the *LD/I* ratio for all $n(n-1)/2$ region profiles, which is simply unfeasible. Instead, we consider only region profiles at the granularity of cells (i.e., regions that correspond to a sequence of adjacent segments) and estimate for each such region the *LD/I* ratio. The estimated contour, $\hat{\mathbb{P}}$, is constructed from those region profiles for which the estimated *LD/I* ratio is $\geq$0.95. Given that $k \ll n$, the number of region profiles to consider is significantly smaller, namely $k(k-1)/2$.

To obtain the numbers of sampled SNP pairs in a region, we have to aggregate the numbers over all cells that make up the corresponding region profile. Let $R_{i,j}$ be a region that is composed of a sequence of consecutive segments, starting with segment x and ending with segment y, and let $\mathsf{M}_{k\times k}$ be the sampling matrix. The aggregated results of the sampling over the region profile $P_{i,j}$ is computed as

$$\mathbf{m}_{i,j} = (\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3) = \sum_{x \leq x' \leq y' \leq y} m_{x',y'}.$$

Example 7. Consider Figure 5.4(a), where a chromosome of 80 SNPs was split into $k = 8$ segments of $l = 10$ SNPs each. The region $R_{11,30} = \langle s_{11}, \ldots, s_{30} \rangle$ is at the granularity of segments since it starts with segment $x = 2$ and ends with segment $y = 3$. The corresponding profile, $P_{11,30}$, covers the cells $C_{2,2}$, $C_{2,3}$ and $C_{3,3}$. The aggregated sampling results in region $R_{11,30}$ are computed as $\mathbf{m}_{11,30} = (0,4,0) + (0,8,2) + (4,0,0) = (4,12,2)$, i.e., 4 *strong LD*, 12 *strong EHR* and 2 *non-informative* pairs (cf. Figure 5.3(b) for the sampling results).

The aggregated numbers of sampled SNP pairs, $\mathbf{m}_{i,j} = (\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3)$, in a region $R_{i,j}$ follow a multinomial distribution with the following parameters: the number of samples, $m = \mathbf{m}_1 + \mathbf{m}_2 + \mathbf{m}_3$, and the true proportions of *strong LD*, *strong EHR*, and *non-informative* SNP pairs, $(\pi^{(1)}_{i,j}, \pi^{(2)}_{i,j}, \pi^{(3)}_{i,j})$, which are given as

$$\pi^{(1)}_{i,j} = \frac{|\{(s_{i'}, s_{j'}) \in P_{i,j} \mid t_{i',j'} = 1\}|}{|P_{i,j}|},$$
$$\pi^{(2)}_{i,j} = \frac{|\{(s_{i'}, s_{j'}) \in P_{i,j} \mid t_{i',j'} = 0\}|}{|P_{i,j}|},$$
$$\pi^{(3)}_{i,j} = \frac{|\{(s_{i'}, s_{j'}) \in P_{i,j} \mid t_{i',j'} = \infty\}|}{|P_{i,j}|}.$$

Since these (true) proportions are not known, the multinomial distribution has three unknown parameters, for which we estimate their confidence intervals. Given the maximum likelihood estimators $\hat{\pi}^{(1)}_{i,j} = \mathbf{m}_1/m$,

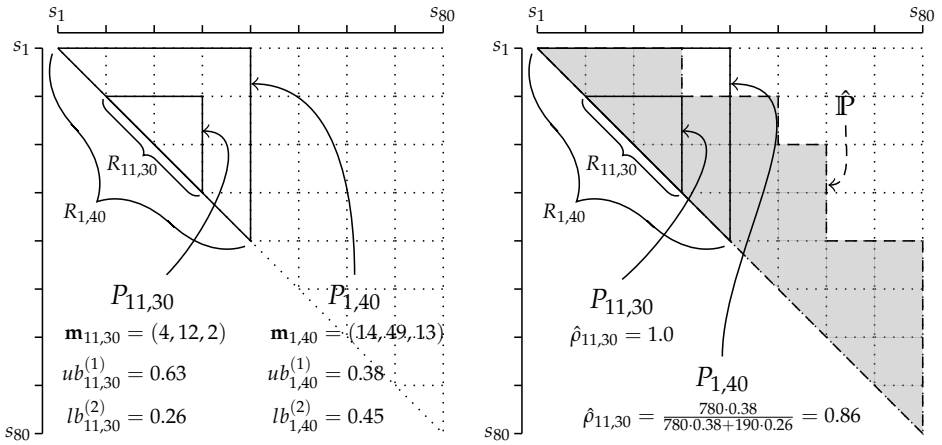


Figure 5.4: Estimating the haplotype block contour.

$\hat{\pi}^{(2)}_{i,j} = \mathbf{m}_2/m$, and $\hat{\pi}^{(3)}_{i,j} = \mathbf{m}_3/m$, the simultaneous $100(1-\alpha)\%$ confidence intervals $[lb^{(v)}_{i,j}, ub^{(v)}_{i,j}] \subseteq [0,1]$, where $v \in \{1,2,3\}$, for every proportion $\pi^{(v)}_{i,j}$ can be obtained using the Fitzpatrick and Scott method [42]:

$$lb^{(v)}_{i,j} = \hat{\pi}^{(v)}_{i,j} - \frac{z_{\alpha/2}}{2\sqrt{m}},$$
$$ub^{(v)}_{i,j} = \hat{\pi}^{(v)}_{i,j} + \frac{z_{\alpha/2}}{2\sqrt{m}},$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)\%$ percentile of the standard normal distribution and α is the probability of a type I error. The Quesenberry and Hurst (QH) [43] with Sison and Glaz (SG) [44] methods for computing simultaneous confidence intervals were also considered in the experiments. However, they failed to support the nominal confidence level for extreme proportions and very small samples (cf. Section 5.4).

Example 8. In Figure 5.4(a), the aggregated sampling result over the profile $P_{11,30}$ is $\mathbf{m}_{11,30} = (4, 12, 2)$, i.e., 4 *strong LD*, 12 *strong EHR* and 2 *non-informative* pairs. Thus, the total number of samples in the region is $m = 4 + 12 + 2 = 18$, and the maximum likelihood estimator for *strong LD* pairs is $\hat{\pi}^{(1)}_{i,j} = 4/18$. Assume $\alpha = 0.00056$, corresponding to a $\approx$99.9% confidence interval. Given that $z_{0.00028} = 3.45$, the upper bound of the 99.9% confidence interval for the proportion $\pi^{(1)}_{11,33}$ of *strong LD* pairs is

$$ub^{(1)}_{11,30} = \frac{4}{18} + \frac{3.45}{2\sqrt{18}} \approx 0.63.$$

The lower bound as well as the bounds for the *strong EHR* pairs are computed in a similar way.

Using the above confidence intervals, we can estimate the *LD/I* ratio of *strong LD* to *non-informative* pairs in a given region.

Definition 11. (*Estimated LD/I Ratio*). Let $R_{i,j}$ be a region at the level of segments with profile $P_{i,j}$, α be the type I error probability, $[lb^{(1)}_{i,j}, ub^{(1)}_{i,j}]$ be the $100(1-2\alpha)\%$ confidence interval of the proportion $\pi^{(1)}_{i,j}$, and $[lb^{(2)}_{i+l,j-l}, ub^{(2)}_{i+l,j-l}]$ be the $100(1-2\alpha)\%$ confidence interval of the proportion $\pi^{(2)}_{i+l,j-l}$. The estimated *LD/I* ratio, $\hat{\rho}_{i,j}$, is defined as

$$\hat{\rho}_{i,j} = \begin{cases} 1 & j - i \leq 2l, \\ \frac{|P_{i,j}| \cdot ub^{(1)}_{i,j}}{|P_{i,j}| \cdot ub^{(1)}_{i,j} + |P_{i+l,j-l}| \cdot lb^{(2)}_{i+l,j-l}} & \text{otherwise.} \end{cases}$$

For regions that are constituted only of one segment or two segments, i.e., $j - i \leq 2l$, we set the estimated *LD/I* ratio to 1. This is to avoid that very short haplotype blocks formed by nearby SNPs are missed. Indeed, even if a very low proportion of *strong LD* pairs is sampled in a near-diagonal cell, their concentration near the diagonal could still be sufficient to form short haplotype blocks, since nearby SNPs are more likely to be highly associated. In all other cases, we compute a very conservative estimator of the *LD/I* ratio, by assuming the maximal number of *strong LD* pairs (i.e., the upper bound of the corresponding confidence interval) and the minimal number of *strong EHR* pairs (i.e., lower bound of the corresponding confidence interval).

Definition 12. (*Estimated Haplotype Block Contour*). Let chromosome $S = \langle s_1, \ldots, s_n \rangle$ be split into k, $1 < k < n$, adjacent segments of $l = \frac{n}{k}$ SNPs each. The *estimated haplotype block contour* is defined as

$$\hat{\mathbb{P}} = \bigcup_{\substack{1 \leq i < j \leq n:\ i \bmod l = 1 \wedge \\ j \bmod l = 0 \wedge \\ \hat{\rho}_{i,j} \geq 0.95}} P_{i,j}.$$

For the estimated haplotype block contour we consider only regions at the granularity of segments, i.e., regions that start with a segment ($i \mod l = 1$) and end with a segment ($j \mod l = 0$). If the estimated *LD/I* ratio is ≥ 0.95, all SNP pairs in the corresponding region profile, $P_{i,j}$, are added to the contour.

Example 9. Consider the regions $R_{11,30}$ and $R_{1,40}$ in Figure 5.4(b). The number of SNP pairs in the corresponding region profiles are $|P_{11,30}| = 190$ and $|P_{1,40}| = 780$. Since $30 - 11 \leq 2 \cdot 10$, the $P_{11,30}$ profile is constituted only from near-diagonal cells and the estimated *LD/I* ratio in the corresponding $R_{11,30}$ region is set to $\hat{\rho}_{11,30} = 1.0$. For the $R_{1,40}$, the estimated

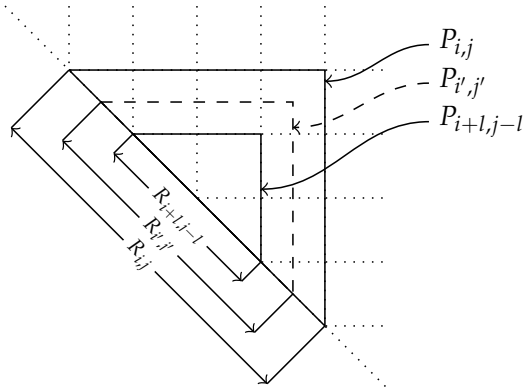


Figure 5.5: Inner region $R_{i+l,i-l}$ and outer region $R_{i,j}$.

LD/I ratio is computed as $\hat{\rho}_{1,40} = \frac{780 \cdot 0.38}{780 \cdot 0.38 + 190 \cdot 0.26} = 0.86$. The $P_{11,30}$ profile with $\hat{\rho}_{11,30} = 1 > 0.95$ is added to the estimated haplotype block contour, whereas $P_{1,40}$ with $\hat{\rho}_{1,40} = 0.86 < 0.95$ is not. The grey color in Figure 5.4(b) marks the estimated haplotype block contour.

5.2.5 Properties

Definition 11 requires to specify a priori for every region profile the type I error probability, α, for the simultaneous confidence intervals of the proportions $\pi^{(1)}$ and $\pi^{(2)}$. Typically, the probability of type I error is set to 1%, i.e., allowing 1 incorrect result in 100. In our case, α is controlled through the pre-specified probability p of obtaining an estimated contour that covers the true contour. Precisely, α is derived from p by adjusting it for the multiple tests according to the Šidák correction method [45], i.e., $\alpha = 1 - p^{2/(k(k+1))}$. This multiple testing correction is known to be conservative. Such conservative behavior is essential, since it guarantees the pre-specified probability p, which is typically set to 0.95 or 0.99.

We use the estimated *LD/I* ratio in Definition 11 to decide if a region profile should be included in the estimated haplotype block contour or not. We can prove that, if the estimated *LD/I* ratio is <0.95, the inclusion of the region profile to the estimated haplotype block contour will most likely not cover any new haplotype blocks.

Theorem 2 *Let $R_{i,j}$ be a region with $\hat{\rho}_{i,j} < 0.95$ and $i - j > 2l$. Then, with probability $\geq 100(1-\alpha)\%$ there exists no haplotype block $R_{i',j'} = \langle s_{i'}, \ldots, s_{j'} \rangle$ with profile $P_{i',j'}$ such that $P_{i+l,j-l} \subset P_{i',j'} \subseteq P_{i,j}$.*

Proof. We do a proof by contradiction and assume that there exists a haplotype block $R_{i',j'} = \langle s_{i'}, \ldots, s_{j'} \rangle$ such that $P_{i+l,j-l} \subset P_{i',j'} \subseteq P_{i,j}$ (cf. Figure 5.5). By Definition 8, $\rho_{i',j'} = n_1/(n_1 + n_2) \geq 0.95$, where n_1 and n_2

are, respectively, the number of *strong LD* and *strong EHR* pairs in $R_{i',j'}$. Let n'_2 be the number of *strong EHR* pairs in region $R_{i+l,i-l}$. Since the region profile $P_{i',j'}$ completely covers $P_{i+l,i-l}$, we have $n'_2 \leq n_2$. Let n'_1 be the number of *strong LD* pairs in $R_{i,j}$. Since the region profile $P_{i',j'}$ is completely covered by $P_{i,j}$, we get $n'_1 \geq n_1$. Provided that the estimation of the simultaneous confidence interval bounds $ub^{(1)}_{i,j}$ and $lb^{(2)}_{i+l,j-l}$ is correct, the inequalities $|P_{i,j}| \cdot ub^{(1)}_{i,j} \geq n'_1 \geq n_1$ and $|P_{i+l,j-l}| \cdot lb^{(2)}_{i+l,j-l} \leq n'_2 \leq n_2$ hold. From this we can derive $\rho_{i',j'} = n_1/(n_1+n_2) \leq n'_1/(n'_1+n'_2) \leq |P_{i,j}| \cdot ub^{(1)}_{i,j} / (|P_{i,j}| \cdot ub^{(1)}_{i,j} + |P_{i+l,j-l}| \cdot lb^{(2)}_{i+l,j-l})$, and further $\rho_{i',j'} \leq \hat{\rho}_{i,j} < 0.95$, which is a contradiction to our initial assumption $\rho_{i',j'} \geq 0.95$. This contradiction holds under the assumption of correctly estimated simultaneous confidence interval bounds, $ub^{(1)}_{i,j}$ and $lb^{(2)}_{i+l,j-l}$, which is the case with probability $\geq 100(1-\alpha)\%$. Note, that in some cases the contradiction may hold even for incorrectly estimated $ub^{(1)}_{i,j}$ and $lb^{(2)}_{i+l,j-l}$, e.g., when the error in the estimate is too low to significantly affect the value of the estimated *LD/I* ratio $\hat{\rho}_{i,j}$. This proves the theorem. ■

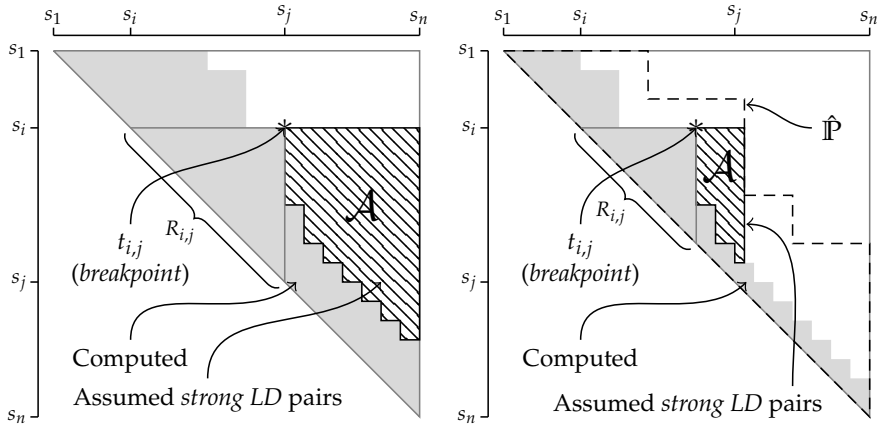
Theorem 2 uses two nested regions at the granularity of segments: an outer region $R_{i,j}$ and an inner region $R_{i+l,i-l}$. If the estimated *LD/I* ratio in the outer region is less than 0.95, the probability to find a region that forms a haplotype block and is longer than $R_{i+l,j-l} = \langle s_{i+l}, \ldots, s_{j-l} \rangle$ and shorter than $R_{i,j} = \langle s_i, \ldots, s_j \rangle$ is very low. For this reason, the profile of the outer region is not included into $\hat{\mathbb{P}}$, as it doesn't help to identify additional haplotype blocks.

Using the result from Theorem 2 and the Šidák correction method, we can prove that the estimated haplotype block contour $\hat{\mathbb{P}}$ contains all haplotype blocks with probability $\geq p$.

Theorem 3 *Let $S = \langle s_1, \ldots, s_n \rangle$ be a chromosome with true haplotype block contour $\mathbb{P}$. For the estimated haplotype block contour, $\hat{\mathbb{P}}$, and a pre-specified probability p, $0 < p < 1$, the following holds:*

$$\mathsf{P}(\mathbb{P} \subseteq \hat{\mathbb{P}}) \geq p.$$

Proof. Suppose that $\mathbb{P} \nsubseteq \hat{\mathbb{P}}$. Then, at the granularity of segments, there must exist at least one region $R_{i,j}$ with $i - j > 2l$ and $\hat{\rho}_{i,j} < 0.95$ and one region $R_{i',j'}$ with $\rho_{i',j'} \geq 0.95$ such that $P_{i+l,j-l} \subset P_{i',j'} \subseteq P_{i,j} \wedge P_{i+l,j-l} \subset \hat{\mathbb{P}} \wedge P_{i,j} \not\subset \hat{\mathbb{P}}$ (cf. Figure 5.5). From Theorem 2 we have that the probability that there exists at least one such region $R_{i,j}$ out of all $k(k+1)/2$ regions is less than $1-(1-\alpha)^{k(k+1)/2}$. Given that $\alpha = 1 - p^{2/k(k+1)}$ after the Šidák correction, we obtain $1-(1-\alpha)^{k(k+1)/2} = 1-(1-(1-p^{2/k(k+1)}))^{k(k+1)/2} =$



(a) Original MIG^{++} (b) Modified MIG^{++}

Figure 5.6: Schematic representations of the original and modified MIG^{++} algorithms.

$1 - (p^{2/k(k+1)}))^{k(k+1)/2} = 1 - p$. Thus, the probability of $\mathbb{P} \nsubseteq \hat{\mathbb{P}}$ is less than $1 - p$, from which we get that the probability of $\mathbb{P} \subseteq \hat{\mathbb{P}}$ must be greater or equal to p. ■

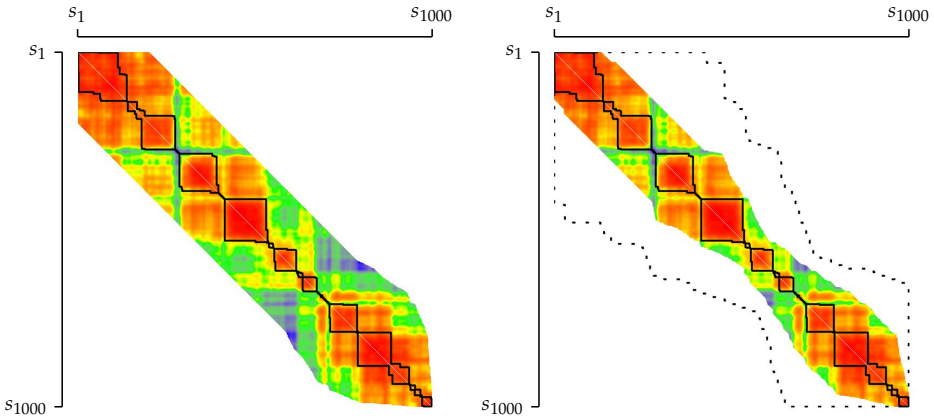
By setting the probability p to 0.99, Theorem 3 provides a strong theoretical support that all true haplotype blocks will be recognized in at least 99% of the runs.

5.3 Haplotype Blocks Refinement

The haplotype blocks refinement step uses the estimated contour for search space pruning. More specifically, during the refinement of the exact haplotype blocks, only the LD matrix elements inside the estimated contour are considered, whereas the rest of the LD matrix is omitted. For the refinement step, either the Haploview algorithm or the MIG^{++} algorithm can be used. Since the latter significantly outperforms the Haploview algorithm both in terms of runtime and memory usage, we adopt the MIG^{++} algorithm to account for the estimated haplotype block contour.

Figure 5.6(a) illustrates the MIG^{++} algorithm[1]. It computes the LD matrix starting from the near-diagonal elements towards the top-right element. The gray area denotes the LD matrix elements computed so far. After computing a matrix element $t_{i,j}$, the algorithm checks if the corresponding region $R_{i,j}$ satisfies the haplotype block definition. If this is the case, the region is added to the result. Additionally, the algorithm checks if any of the extended regions, $R_{i,j+1}, \ldots, R_{i,n}$, may possibly be a haplotype

[1]For the more detailed description of MIG^{++} see Chapter 4, Section 4.4.



(a) Original MIG^{++} (b) Modified MIG^{++}

Figure 5.7: The LD matrix of chr20:14,759,169-15,028,962 in the 1000 Genomes Project data computed by the original and modified MIG^{++} algorithms.

block. For this it is assumed that all SNP pairs inside these extended regions for which the LD coefficient is not yet computed are *strong LD* pairs. In Figure 5.6(a), the hatched area $\mathcal{A}$ marks all LD matrix elements that are assumed to be *strong LD* pairs. If under such a conservative assumption there is no evidence of a possible haplotype block, a breakpoint is set at $t_{i,j}$, and the elements $t_{i,j+1}, \ldots, t_{i,n}$ are omitted from the next computational steps.

Figure 5.6(b) illustrates the modified MIG^{++} algorithm that accounts for the estimated haplotype block contour. The area $\mathcal{A}$ of assumed *strong LD* pairs is now bounded by the estimated haplotype block contour $\hat{\mathbb{P}}$ and is therefore significantly smaller. As a consequence, breakpoints are recognized much earlier than in the original version, hence more computations are pruned.

Figures 5.7(a) and 5.7(b) show an example of the LD matrix for a chromosomal region of 1,000 SNPs computed by the original and the modified MIG^{++} algorithms, respectively. The colors red, green and blue correspond to strong, moderate and weak LD, respectively. The black line indicates the true haplotypes block contour, $\mathbb{P}$. The dashed line indicates the estimated contour, $\hat{\mathbb{P}}$, which was used by the modified MIG^{++} algorithm. The use of the estimated haplotype block contour in Figure 5.7(b) allowed a more effective search space pruning without affecting the correctness of the result. Note that the difference between the two algorithms is much more evident for longer chromosomal regions.

Since MIG^{++} always computes a correct result, the new sampling-based algorithm S-MIG^{++} computes a correct result if and only if the es-

Region	Start (bp)	End (bp)	ρ	$\lvert\mathbb{P}\rvert$	η
1	225,444	1,223,234	0.005	51,609	0.004
2	17,706,656	18,706,804	0.019	120,088	0.010
3	24,520,185	25,828,521	0.056	363,643	0.029

Table 5.1: The three regions of 5,000 SNPs selected for the experiments in chromosome 20 from the 1000 Genomes Project data. ρ is the *LD/I* ratio in a region, $|\mathbb{P}|$ is the number of SNP pairs inside the haplotype block contour, and η is the proportion of the LD matrix elements covered by the contour.

timated haplotype block contour covers the true haplotype block contour, which happens with a very high probability (cf. Theorem 3).

5.4 Experiments

Experiments were performed on the phased CEPH genotypes in chromosome 20 from the 1000 Genomes Project phase 1 release 3 dataset [24]. The data included 243,080 non-monomorphic SNPs from 170 haplotypes (85 individuals).

The confidence intervals of $|D'|$ were estimated using the Wall and Pritchard method [20] based on a likelihood function evaluated on a 1,000 points grid. If not stated otherwise, the probability to obtain all true haplotype blocks is set to $p = 0.99$.

We assessed the error rate of S-MIG^{++} and the influence of different choices for k and σ on genomic regions with different characteristics. Specifically, we scanned the entire chromosome 20 with a sliding window of 5,000 consecutive SNPs. For each sequence of 5,000 SNPs, the *LD/I* ratio was computed. Then, we selected three non-overlapping regions such that the *LD/I* ratio was minimal, average and maximal. For each of the selected regions, the true haplotype block contour $\mathbb{P}$ was computed. Table 5.1 lists the three selected regions.

To assess runtime and memory usage depending on the number of SNPs, we selected a region in the middle of chromosome 20. Then, we gradually expanded the region by adding a fixed number of SNPs to the left and to the right each time, until the entire chromosome was covered.

All experiments were run on a machine with an Intel Xeon E5-4640 (2.4 GHz) CPU. The C++ sources of S-MIG^{++} were compiled using version 4.9.1 of the GNU Compiler Collection (GCC). We also implemented a parallel version of S-MIG^{++} using Open MPI 1.6.5. The runtime of the parallelized S-MIG^{++} was evaluated with disabled hyper-threading in a single CPU core.

5.4.1 Error Rate

The most crucial part of the haplotype block contour estimation is the computation of the simultaneous confidence intervals for the proportions of *strong LD* and *strong EHR* SNP pairs within the region profiles. The incorrect estimation of these proportions may lead to the incorrect estimation of the *LD/I* ratio. We assessed three methods for the computation of simultaneous confidence intervals for multinomial proportions: (i) Quesenberry and Hurst (QH) [43]; (ii) Fitzpatrick and Scott (FS) [42]; (iii) Sison and Glaz (SG) [44]. The empirical coverage probability of each method was computed by counting how often in 10,000 simulations the true proportion $\pi^{(1)}$ of *strong LD* pairs was below its estimated upper bound $ub^{(1)}$ and how often the true proportion $\pi^{(2)}$ of *strong EHR* pairs was above its estimated lower bound $lb^{(2)}$.

Figure 5.8 shows the minimal empirical coverage probability for the three methods when the nominal coverage probability was set to $p = 0.99$. All possible regions of 15, 45 and 90 SNPs with their corresponding profiles containing 105, 990 and 4,005 SNP pairs were considered. The x-axis shows the product $\pi^{(1)} \cdot \pi^{(2)} \cdot \pi^{(3)}$ between the true proportions of *strong LD*, *strong EHR* and *non-informative* SNP pairs inside a region profile P. This product was close to 0 if at least one proportion was very low, and was close to its maximum of 0.037 when all three proportions were close to each other (i.e., close to $1/3$). For the simplicity and better readability of the plot, we subdivided the product $\pi^{(1)} \cdot \pi^{(2)} \cdot \pi^{(3)}$ into categories. The y-axis shows the minimal empirical coverage probability for every different category of the product $\pi^{(1)} \cdot \pi^{(2)} \cdot \pi^{(3)}$. The minimal empirical coverage probability improved for all methods along with the increase of the sampling proportion σ and the size of the region profile $|P|$. However, in cases of extreme proportions (i.e., when $\pi^{(1)} \cdot \pi^{(2)} \cdot \pi^{(3)}$ was close to 0), the QH and SG methods had an empirical coverage probability that is much below the nominal level, even for large σ or large $|P|$. In contrast, the FS method showed a stable behavior over all scenarios.

Figure 5.9 shows the empirical probability of all correctly estimated simultaneous confidence intervals when running the S-MIG^{++} algorithm with nominal probability $p = 0.99$. The x-axis represents the number of segments k, while the y-axis represents the empirical probability of all correctly estimated simultaneous confidence intervals. We run the algorithm 1,000 times for every combination of k and σ. Under the multiple-testing correction, we would expect that the empirical probability p never falls below 0.99. However, the failure of the QH and SG methods to support the nominal coverage probability of simultaneous confidence intervals resulted in a failure to support the nominal probability p. The nominal probability was not reached with the QH and SG methods, even when the number of segments k was relatively small and the sampling fraction σ was

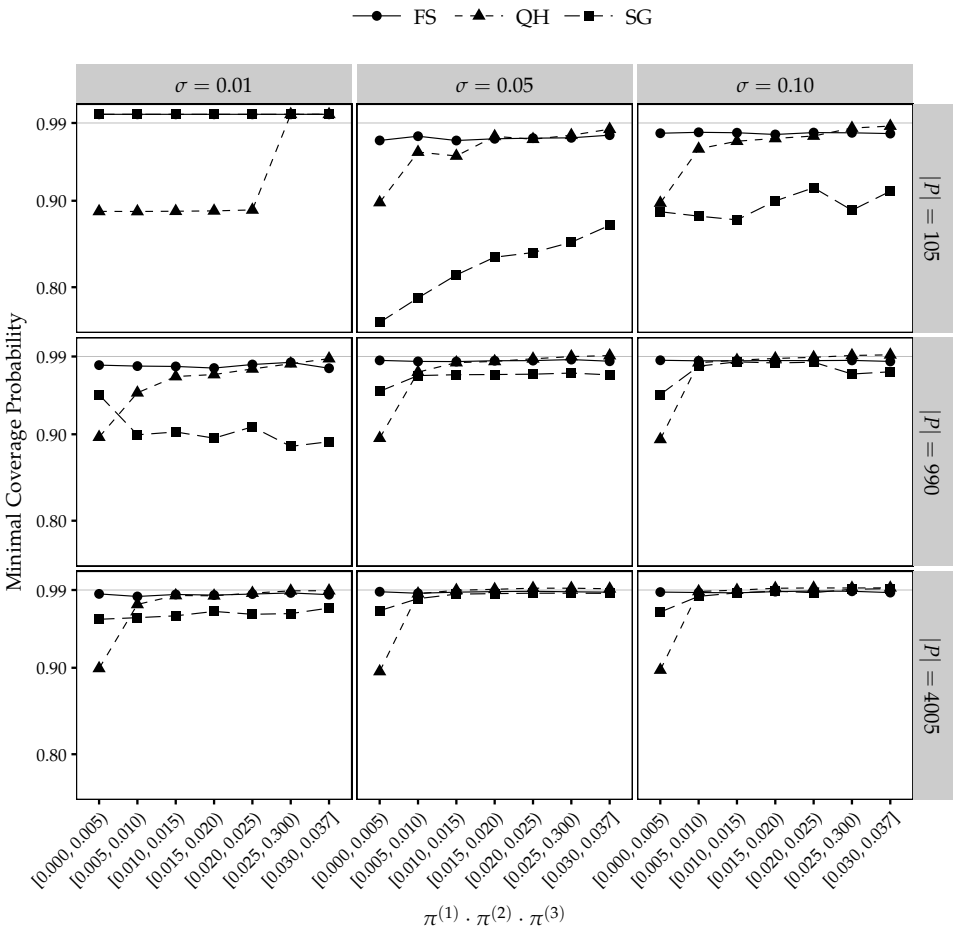


Figure 5.8: The minimal coverage probability for the QH, FS and SG methods in region profiles of different size $|P|$ and using different sampling fractions σ at the nominal coverage probability level of 0.99.

relatively large. In contrast, for the FS method the empirical probability was always above the nominal $p = 0.99$.

Based on the results of these experiments, the S-MIG^{++} was implemented with the FS method, since it better guarantees the validity of Theorems 2 and 3.

5.4.2 Precision of the Estimated Haplotype Block Contour

Figure 5.10 shows the dependency of the size of the estimated haplotype block contour on the number of segments k and the sampling fraction σ. The x-axis represents the number of segments k, while the y-axis represents the proportion $\hat{\eta}$ of LD matrix elements that are covered by the estimated

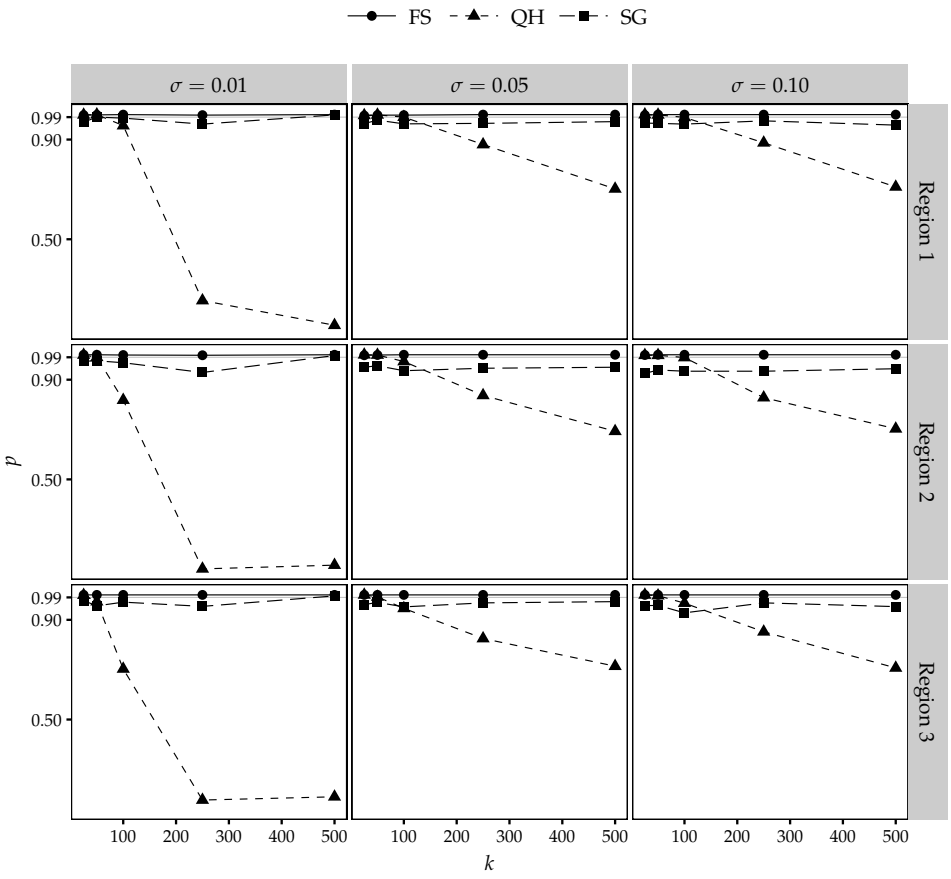


Figure 5.9: The empirical probability of all correctly estimated simultaneous confidence interval bounds when running the S-MIG^{++} algorithm with a nominal probability $p = 0.99$ and using different sampling fraction σ.

haplotype block contour. An estimated haplotype block contour which covers lowest proportion $\hat{\eta}$ of LD matrix elements is considered more precise. With all three different true haplotype block contours, the smallest $\hat{\eta}$ was achieved with the highest number of segments $k = 500$ and the highest sampling fraction $\sigma = 0.20$.

Figure 5.11 illustrates the effect of varying k and σ on the estimated haplotype block contour. The true haplotype block contour is colored in black. Figure 5.11(a) shows the estimated haplotype block contours with different numbers of segments k for a fixed sampling fraction $\sigma = 0.20$, while Figure 5.11(b) shows the estimated haplotype block contours with different sampling fractions σ for a fixed number of segments $k = 500$. By increasing the sampling fraction σ, the available LD information grows. This yields a more accurate estimated contour, even when k is small. On

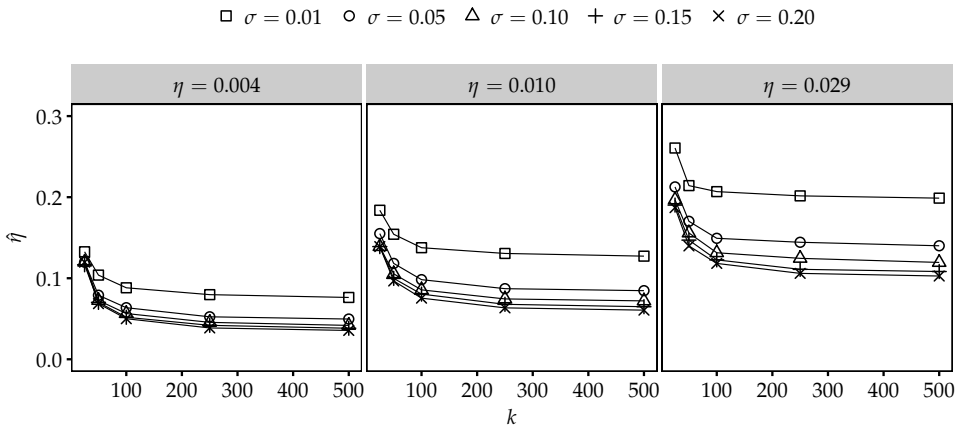
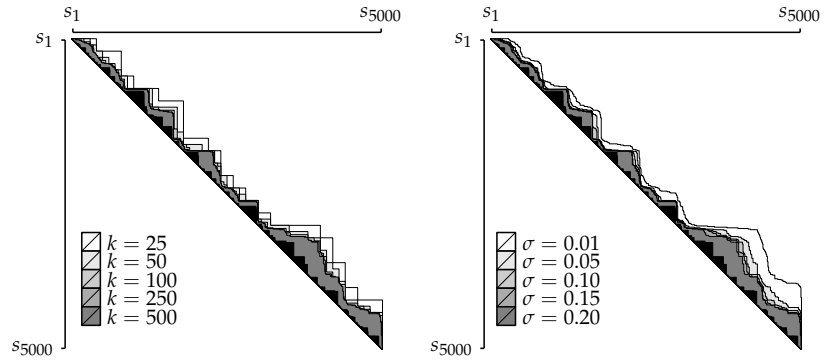


Figure 5.10: Size of the estimated haplotype block contour for different values of k, σ and η.



(a) Fixed sampling fraction $\sigma = 0.20$ (b) Fixed number of segments $k = 500$

Figure 5.11: The estimated haplotype block contours in chr20:24,520,185-25,828,521 with 5,000 SNPs at different values of k and σ.

the other hand, the larger the number of segments k, the finer the partition, and so redundant LD matrix elements can more easily be omitted.

Relying on the empirical results that for a fixed sampling fraction σ the most precise contour is obtained with the largest number of segments k, we propose to determine k as follows:

$$k = \min \left\{ k' \mid 1 \leq \left\lfloor \frac{n}{k'} \right\rfloor^2 \cdot \sigma < 2 \wedge k' \leq n \right\} \tag{5.1}$$

This formula guarantees the minimal cell size that is sufficient to sample at least one LD matrix element. As the sampling fraction increases, the cell size decreases. The number of samples for the computation of simultane-

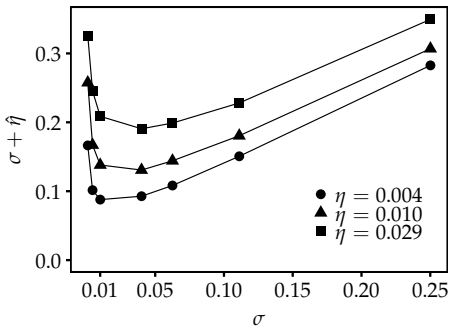


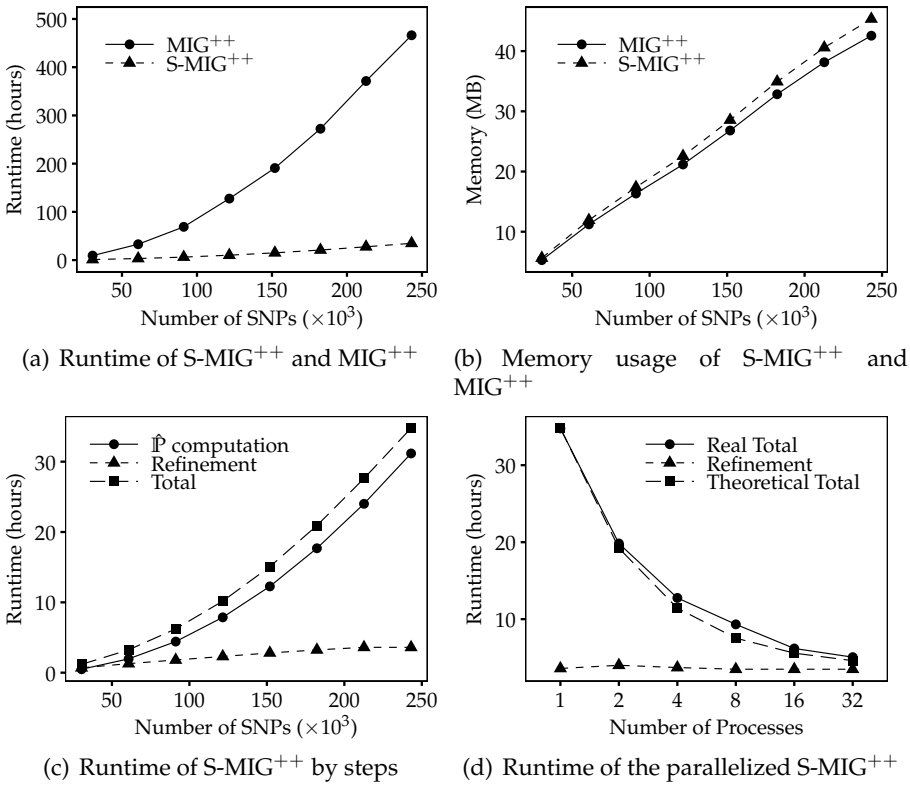
Figure 5.12: The sum of sampled SNP pairs and SNP pairs covered by the estimated haplotype block contour.

ous confidence intervals is always increasing when more cells are aggregated together into profiles. Minimizing the cell size increases the number of computations of simultaneous confidence intervals with the FS method. Since the FS method is computationally very cheap and has constant runtime complexity with respect to the number of samples, we assume its cost to be equal to 0.

Although an increase of the sampling fraction σ improves the precision of the estimated contour, it also increases the number of expensive computations of LD coefficients between sampled SNP pairs. To obtain the best runtime performance, it is important to minimize the sum of sampled SNP pairs and SNP pairs covered by the estimated haplotype block contour. Figure 5.12 shows the dependency of this sum on the sampling fraction. The x-axis represents the sampling fraction σ, while the y-axis represents the sum of σ and the proportion $\hat{\eta}$ of the LD matrix elements covered by the estimated haplotype block contour. For the three different regions in Table 5.1 of 5,000 SNPs each, the optimal σ was between 0.01 and 0.05.

5.4.3 Runtime and Memory Usage

The S-MIG^{++} algorithm significantly outperformed MIG^{++} in terms of runtime. For 250,000 SNPs, the total calculation time was 34.8 hours with S-MIG^{++} and 466.2 hours with MIG^{++} (Figure 5.13(a)). Due to the maintenance of the estimated block contour, S-MIG^{++} requires slightly more memory than MIG^{++} (Figure 5.13(b)). However, thanks to an incremental computation of the $\mathsf{M}_{k\times k}$ sampling matrix, the memory complexity of S-MIG^{++} is linear in the size of the data. By breaking down the runtime complexity of S-MIG^{++}, we noticed a quadratic runtime for the haplotype block contour estimation and a linear runtime for the refinement step (Figure 5.13(c)). The quadratic complexity of the estimation step is due to the quadratically growing number of sampled SNP pairs. More specifically,



(a) Runtime of S-MIG^{++} and MIG^{++}

(b) Memory usage of S-MIG^{++} and MIG^{++}

(c) Runtime of S-MIG^{++} by steps

(d) Runtime of the parallelized S-MIG^{++}

Figure 5.13: Performance of the S-MIG^{++} algorithm.

the runtime complexity of the estimation step is equal to $O(n^{2+\frac{\log_{10}\sigma}{\log_{10}n}})$. Assuming a sampling fraction $\sigma = 0.01$, the complexity would be equal to $O(n^{2(1-\frac{1}{\log_{10}n})})$. If we further assume that the longest haplotype block has h SNPs, the size of the haplotype block contour is bounded by $h \cdot n$ from above. Thus, the refinement step has $O(\hat{h} \cdot n)$ runtime complexity, where $\hat{h}$ is the estimated number of SNPs in the longest haplotype block.

5.4.4 Parallelized Contour Estimation

Experiments showed that the contour estimation is the most expensive step in the S-MIG^{++} algorithm. The runtime of this step can be further reduced by parallelization. Since the sampling within a cell is totally independent from the sampling in any other cell, the cells can be processed in parallel.

The main process distributes the coordinates x and y of the cell $C_{x,y}$ between all the processes (including itself). Once a process receives the coordinates x and y, it samples from the cell $C_{x,y}$ and sends the sampling results $m_{x,y}$ back to the main process. The main process received all the sampling results and estimates the *LD/I* ratio for the corresponding regions and their

profiles. The cells are repeatedly distributed between all the processes in consecutive chunks of equal size until no unprocessed cell remains.

Figure 5.13(d) shows the runtime of the parallelized S-MIG^{++} algorithm when applied to the entire chromosome 20 (243,080 SNPs). The non-parallel refinement step took 3.6 hours in average. With 32 parallel processes, the haplotype blocks recognition took only 5.1 hours, while with only one process it took 34.8 hours. The experimental speedup was slightly lower than the theoretically expected, which we attribute to the cost of data exchange between processes and their management. Note that the efficiency of the parallelization may be affected by the specific hardware configuration. Our main aim in this experiment was to demonstrate the feasibility of S-MIG^{++} parallelization with the potential for further runtime improvements.

5.5 Summary

The S-MIG^{++} algorithm is significantly faster than the MIG^{++} algorithm and the original Haploview [14] algorithm, reducing runtime requirements at the chromosome level from weeks to hours. Precisely, 2.8 weeks using MIG^{++} vs. 34.8 hours using S-MIG^{++} for chromosome 20 with 243,080 SNPs. This speed-up is achieved by randomly sampling SNP pairs to obtain an estimate of upper limits for haplotype blocks boundaries, which are then refined in a second step. We showed that we can apply sampling theory to dramatically reduce the number of computations while keeping the probability of error nearly absent ($<$1%). This was demonstrated through both a theoretical proof and empirical simulations.

By extensive experiments, we observed that in order to estimate reliable upper limits of the true haplotype blocks boundaries in step one, it is sufficient to sample only 1% of all SNP pairs from the entire chromosome. We demonstrate also how the sampling, which is the most demanding task, can be further improved by parallelization ($\approx$5 hours for chromosome 20).

The constantly improving technologies will allow genetic studies to genotype millions of SNPs in hundred thousands of samples. The most recent example is the initiative of UK Biobank to genotype $\approx$850,000 genetic markers in 500,000 recruited participants [46]. The significantly improved scalability of the S-MIG^{++} algorithm over state-of-the-art solutions makes it possible to mine LD patterns in such huge datasets, which was previously infeasible.

Chapter 6

Real Data Application

The runtime efficiency and low memory usage of our algorithms allow us to run the haplotype block partitioning on the entire human genome while considering even long range LD between SNPs. One of many possible applications of the whole-genome haplotype block partition is the haplotype block association scan. In this chapter, we use MIG^{++} on real dataset from North American Rheumatoid Arthritis Consortium (NARAC) to perform haplotype block association scan in a very short time. We assess the difference between haplotype block association and single-SNP association results, and discuss the impact of the WP and AV methods for $|D'_{i,j}|$ CI computation.

6.1 The NARAC Dataset

The NARAC data consisted of 868 cases and 1,194 controls. The samples were genotyped at 544,917 autosomal and sex chromosome SNPs. Quality check was performed with PLINK 1.07 [15]: we excluded 5,422 SNPs with a call rate of <90%, 11,327 SNPs with a minor allele frequency of <0.001, and 898 SNPs because of significant deviation from Hardy-Weinberg equilibrium in controls (*p-value* $\leq 10^{-6}$). No samples were excluded because of low call rate (<90%); 2 cases and 5 controls were removed because of sex mismatch; 1 case and 8 controls were additionally excluded after population stratification test based on principal component analysis performed with EIGENSOFT 5.0.1 [47]. After the quality control, 514,539 autosomal SNPs and 2,046 samples were available for analyses.

Haplotypes were phased using SHAPEIT version 2 [48]. To achieve good accuracy, we set 400 conditioning states per SNP. Recombination rates were taken from HapMap phase II build 36 and effective population size was set to 11,418 (as suggested for CEU populations). The estimated haplotypes were submitted to MIG^{++} and processed with the WP and AV methods. We obtained 98,979 WP blocks, covering 445,832 SNPs, with 68,707

singleton SNPs outside of any block. The AV method identified 97,816 blocks, covering 446,170 SNPs, and 68,369 singleton SNPs.

The genome-wide association scan was based on a logistic regression model adjusted for sex and the top 10 eigenvectors obtained from EIGENSOFT 5.0.1 [47]. The association between disease status and individual SNPs or haplotype blocks was tested with a likelihood ratio test using PLINK 1.0.7 [15] with the logistic-genotypic and omnibus options, respectively. Within each block, haplotypes with frequency of <0.01 were collapsed together to preserve power. Singleton SNPs outside blocks were treated as in the SNP-based analysis, therefore producing analogous results. Genomic control (GC) correction was applied to both SNP- and block-based GWAS results. Bonferroni-corrected significance thresholds were set to 2.98×10^{-7} for analysis based on the WP block partition (i.e. 0.05 divided by the sum of 98,979 WP blocks and 68,707 singleton SNPs), 3.01×10^{-7} for the analysis based on the AV method (i.e. 0.05 divided by 97,816 AV blocks plus 68,369 singleton SNPs), and 9.17×10^{-8} (i.e. 0.05 / 514,539) for the individual SNP analysis.

6.2 GWAS Results

After the GWAS, we observed a genomic inflation factor λ of 1.015 for the SNP-based analysis, 1.082 for the AV block-based analysis, and 1.077 for the WP block-based analysis. After GC correction, in the SNP-based analysis, 116 SNPs were genome-wide significant. Of them, 106 were located inside 25 AV blocks and 110 inside 27 WP blocks. From the AV and WP block-based analyses, we observed 29 and 33 genome-wide significant blocks, respectively. Twenty-three of such blocks were the same between the two methods.

The results from the SNP- and block-based analyses are compared in Table 6.1. The first part of the table shows the 20 genome-wide significant loci detected by both SNP- and block-based analyses. In most cases, the AV and WP methods brought to identical results. One exception was the 4^{th} locus, where two adjacent AV blocks including 6 and 14 SNPs, respectively, corresponded to two adjacent WP blocks of 7 and 13 SNPs, respectively. That is, one SNP shifted from one block to another. In terms of significance, results were practically unchanged. A second exception was locus 13, were a block was detected only with the WP but not with the AV method. The last two exceptions were loci number 15 and 19. In both cases, an AV block was split into two WP blocks. The second part of Table 6.1 shows a number of loci that wouldn't have been detected with a SNP-based GWAS, but were uncovered by at least one of the two block partition methods. The AV and WP methods produced similar results. We didn't observe any clear advantage of one method compared to the other. The last section of the table

shows that there was a small number of loci uncovered only by the SNP-based analysis. For these loci, the p-values from the block-based analyses were often close to the significance level, with the exception of the last two loci.

6.3 Summary

To provide an application of a whole-genome haplotype block partition, we analyzed the data from the North American Rheumatoid Arthritis Consortium (NARAC) dataset using both block partitions: the one obtained with the standard WP method and the one obtained with the AV approach. As observed in previous studies [49,50], the GWAS results were dominated by the HLA locus on chromosome 6. However, other loci were identified in other chromosomes. For what concerns the two block partition methods, the results were very similar, suggesting that the AV approach might be a convenient way to run a fast recognition of the haplotype blocks. However, we recognize that ours was an empirical application based on half a million genotyped SNPs. Results might be different in a larger context, such as that of a GWAS based on the 1000 Genomes Project [24] dataset, where the number of AV blocks is expected to be much smaller than the number of WP blocks, and the AV blocks are expected to be much larger than the WP ones.

Our empirical analysis of the NARAC data also confirmed previous observations that SNP- and block-based analyses are complementary to each other [49,51]. In fact, in our analysis some loci were identified only by the single-SNP analysis, other loci were identified only by the haplotype-block analysis, and others by both methods. Thus, genome-wide haplotype association scans are not in competition with standard GWAS. Genome-wide haplotype association scans should be considered as complementary tools that may help to identify loci that could be overlooked by methods based on single-SNP analysis.

We also observed that haplotype blocks may simplify gene annotation. While only one gene, the HLA-DRA [52], which was reported by previous GWASs, was directly implied by a genome-wide significant SNP, four additional previously reported genes were implied by genome-wide significant blocks: the APOM, HLA-DQA1, HLA-DRB1, and HLA-DQA2 genes [52].

		Block				Top SNP	
Locus	Partition method	Chr:start-end	Sign. SNPs #	SNPs #	P-value	Name	P-value
Genome-wide significant blocks that include genome-wide significant SNPs							
1	AV & WP	6:32,055,439-32,182,782	2	15	$\mathbf{2.71 \times 10^{-12}}$	rs2239689	$\mathbf{1.55 \times 10^{-8}}$
2	AV & WP	6:32,204,222-32,259,421	1	14	$\mathbf{1.32 \times 10^{-16}}$	rs3134943	$\mathbf{5.41 \times 10^{-8}}$
3	AV & WP	6:32,317,005-32,319,063	2	3	$\mathbf{1.25 \times 10^{-8}}$	rs412657	$\mathbf{2.17 \times 10^{-10}}$
4	AV	6:32,323,166-32,328,663	4	6	$\mathbf{1.99 \times 10^{-15}}$	rs9267992	$\mathbf{1.14 \times 10^{-11}}$
	AV	6:32,331,236-32,390,832	4	14	$\mathbf{2.34 \times 10^{-33}}$	rs6910071	$\mathbf{1.92 \times 10^{-32}}$
	WP	6:32,323,166-32,331,236	5	7	$\mathbf{1.32 \times 10^{-17}}$	rs3130320	$\mathbf{7.50 \times 10^{-17}}$
	WP	6:32,332,366-32,390,832	3	13	$\mathbf{1.38 \times 10^{-33}}$	rs6910071	$\mathbf{1.92 \times 10^{-32}}$
5	AV & WP	6:32,397,296-32,445,664	13	24	$\mathbf{3.00 \times 10^{-24}}$	rs547077	$\mathbf{2.60 \times 10^{-18}}$
6	AV & WP	6:32,454,772-32,471,794	7	9	$\mathbf{4.88 \times 10^{-27}}$	rs3817973	$\mathbf{5.48 \times 10^{-26}}$
7	AV & WP	6:32,474,399-32,476,065	1	3	$\mathbf{2.00 \times 10^{-23}}$	rs3817963	$\mathbf{8.55 \times 10^{-23}}$
8	AV & WP	6:32,477,466-32,481,676	1	3	$\mathbf{7.75 \times 10^{-21}}$	rs3806156	$\mathbf{9.43 \times 10^{-14}}$
9	AV & WP	6:32,483,951-32,491,086	5	8	$\mathbf{7.93 \times 10^{-38}}$	rs3763312	$\mathbf{7.00 \times 10^{-34}}$
10	AV & WP	6:32,491,201-32,509,057	6	8	$\mathbf{5.10 \times 10^{-41}}$	rs2395163	$\mathbf{6.14 \times 10^{-37}}$
11	AV & WP	6:32,509,195-32,514,320	6	8	$\mathbf{1.80 \times 10^{-37}}$	rs2395175	$\mathbf{1.83 \times 10^{-39}}$
12	AV & WP	6:32,519,501-32,521,295	3	3	$\mathbf{1.79 \times 10^{-27}}$	rs7192	$\mathbf{2.64 \times 10^{-26}}$
13	WP	6:32,522,251-32,535,767	2	2	$\mathbf{1.10 \times 10^{-28}}$	rs9268832	$\mathbf{4.07 \times 10^{-25}}$
14	AV & WP	6:32,541,145-32,713,862	7	12	$\mathbf{2.23 \times 10^{-47}}$	rs660895	$\mathbf{2.60 \times 10^{-45}}$
15	AV	6:32,760,295-32,766,057	2	5	$\mathbf{4.99 \times 10^{-26}}$	rs9275184	$\mathbf{1.49 \times 10^{-18}}$
	WP	6:32,735,692-32,762,692	3	4	$\mathbf{8.33 \times 10^{-35}}$	rs9275184	$\mathbf{1.49 \times 10^{-18}}$
	WP	6:32,763,196-32,766,057	1	3	$\mathbf{3.33 \times 10^{-23}}$	rs7774434	$\mathbf{5.43 \times 10^{-10}}$
16	AV & WP	6:32,766,602-32,785,130	26	37	$\mathbf{1.71 \times 10^{-40}}$	rs9275224	$\mathbf{1.31 \times 10^{-37}}$
17	AV & WP	6:32,786,977-32,790,115	6	10	$\mathbf{9.32 \times 10^{-50}}$	rs9275595	$\mathbf{1.97 \times 10^{-28}}$
18	AV & WP	6:32,792,235-32,827,644	2	23	$\mathbf{2.94 \times 10^{-19}}$	rs3916765	$\mathbf{2.72 \times 10^{-9}}$
19	AV	6:32.912,776-32,912,887	1	2	$\mathbf{1.46 \times 10^{-8}}$	rs3819721	$\mathbf{5.74 \times 10^{-9}}$
	WP	6:32,912,392-32,912,776	1	2	$\mathbf{9.89 \times 10^{-10}}$	rs3819721	$\mathbf{5.74 \times 10^{-9}}$
	WP	6:32,912,887-32,912,912	0	2	$\mathbf{7.18 \times 10^{-11}}$	rs241425	7.53×10^{-4}
20	AV & WP	9:81,662,684-81,666,969	1	2	$\mathbf{2.72 \times 10^{-8}}$	rs7854383	$\mathbf{3.23 \times 10^{-8}}$

Table 6.1: Results from the rheumatoid arthritis GWAS: comparison between AV and WP haplotype blocks and single-SNP analyses (continued on the next page).

		Block				Top SNP	
Locus	Partition method	Chr:start-end	Sign. SNPs #	SNPs #	P-value	Name	P-value
Genome-wide significant blocks with no corresponding genome-wide significant SNPs							
21	AV & WP	2:219,728,763-219,836,597	0	17	$\mathbf{1.57 \times 10^{-7}}$	rs1052483	2.02×10^{-4}
22	AV	6:31,723,146-31,726,740	0	3	3.17×10^{-7}	rs3130050	5.27×10^{-5}
	WP	6:31,723,146-31,726,740	0	3	$\mathbf{2.98 \times 10^{-7}}$	rs3130050	5.27×10^{-5}
23	AV & WP	6:31,728,499-31,777,475	0	10	$\mathbf{4.11 \times 10^{-8}}$	rs2280800	1.28×10^{-4}
24	AV	6:31,910,520-31,953,964	0	10	$\mathbf{8.66 \times 10^{-8}}$	rs9267658	1.22×10^{-5}
	WP	6:31,885,925-31,945,256	0	8	1.08×10^{-5}	rs2075800	3.58×10^{-5}
	WP	6:31,946,420-31,953,964	0	5	1.73×10^{-6}	rs9267658	1.22×10^{-5}
25	AV	6:31,958,311-31,959,213	0	2	6.00×10^{-1}	rs652888	1.48×10^{-2}
	AV	6:31,968,316-32,026,839	0	10	1.53×10^{-5}	rs1042663	7.47×10^{-7}
	WP	6:31,959,213-32,026,839	0	11	$\mathbf{4.59 \times 10^{-8}}$	rs1042663	7.47×10^{-7}
26	AV & WP	6:32,027,809-32,038,441	0	5	$\mathbf{1.43 \times 10^{-9}}$	rs437179	2.20×10^{-5}
27	AV & WP	6:32,262,976-32,263,559	0	2	$\mathbf{8.89 \times 10^{-9}}$	rs204994	1.83×10^{-4}
28	AV & WP	6:32,296,361-32,298,006	0	4	$\mathbf{1.47 \times 10^{-10}}$	rs3132946	1.51×10^{-7}
29	AV & WP	6:32,300,538-32,303,337	0	5	$\mathbf{6.87 \times 10^{-10}}$	rs499691	5.62×10^{-4}
30	AV	6:32,870,369-32,889,502	0	12	$\mathbf{4.33 \times 10^{-10}}$	rs7767167	2.61×10^{-4}
	WP	6:32,871,088-32,889,502	0	11	$\mathbf{3.77 \times 10^{-10}}$	rs7767167	2.61×10^{-4}
31	AV	6:32,912,776-32,912,887	1	2	$\mathbf{1.46 \times 10^{-8}}$	rs3819721	$\mathbf{5.74 \times 10^{-9}}$
	WP	6:32,912,392-32,912,776	1	2	$\mathbf{9.89 \times 10^{-10}}$	rs3819721	$\mathbf{5.74 \times 10^{-9}}$
	WP	6:32,912,887-32,912,912	0	2	$\mathbf{7.18 \times 10^{-11}}$	rs241425	7.53×10^{-4}
32	AV & WP	6:33,012,959-33,069,082	0	21	$\mathbf{8.10 \times 10^{-8}}$	rs3135034	5.13×10^{-5}
Genome-wide significant SNPs with no corresponding genome-wide significant blocks							
33	AV & WP	1:18,189,820-18,200,270	1	8	1.81×10^{-5}	rs16861613	$\mathbf{5.08 \times 10^{-8}}$
34	AV & WP	6:32,307,122-32,313,088	2	5	2.04×10^{-6}	rs9267873	$\mathbf{3.98 \times 10^{-8}}$
35	AV	10:112,614,407-112,822,215	1	23	5.73×10^{-7}	rs3750619	$\mathbf{8.29 \times 10^{-10}}$
	WP	10:112,614,407-112,749,598	0	18	3.37×10^{-7}	rs3750619	$\mathbf{8.29 \times 10^{-10}}$

Table 6.1: Results from the rheumatoid arthritis GWAS: comparison between AV and WP haplotype blocks and single-SNP analyses (continued on the next page).

		Block				Top SNP	
Locus	Partition method	Chr:start-end	Sign. SNPs #	SNPs #	P-value	Name	P-value
	WP	10:112,754,584-112,822,215	0	5	1.89×10^{-1}	rs10787298	1.10×10^{-1}
36	AV & WP	17:34,570,514-34,575,487	1	5	1.40×10^{-1}	rs593772	$\mathbf{5.50 \times 10^{-9}}$
37	AV & WP	17:63,271,780-63,418,511	1	8	3.26×10^{-1}	rs7502707	$\mathbf{2.02 \times 10^{-8}}$

Table 6.1: Results from the rheumatoid arthritis GWAS: comparison between AV and WP haplotype blocks and single-SNP analyses.

Chapter 7

Conclusions and Future Work

7.1 Summary

The main contribution of this thesis is a series of novel runtime and memory efficient algorithms for the recognition of haplotype blocks based on the most commonly used Gabriel *et al.* [13] definition. The proposed novel idea of incremental construction of haplotype blocks in the MIG algorithm improves the memory complexity from quadratic to linear. Meanwhile, the sophisticated search space pruning developed in the MIG^{+} and MIG^{++} algorithms reduces the runtime by >80% compared to the original algorithm implemented in Haploview [14]. In the S-MIG^{++} algorithm, we introduced a novel sampling-based approach which improves the runtime by another order of magnitude compared to MIG^{++}. The use of an approximated $D'_{i,j}$ variance estimator was assessed as an alternative way to improve the runtime.

The efficiency of the introduced algorithms allows running haplotype block recognition on entire human genome without any constraints on the maximal haplotype block size and considering LD between SNPs at any distance, which was impossible previously. The entire HapMap II [40] CEPH dataset was processed by MIG^{++} in 457 hours, while the entire 1000 Genomes Project [24] CEPH dataset was processed in 44 hours using MIG^{++} with the approximated $D'_{i,j}$ variance estimator. The approximated estimator introduced larger haplotypes blocks and coarser partition. The S-MIG^{++} algorithm avoids approximations and processed entire chromosome 20 from 1000 Genomes Project [24] CEPH dataset with 243,080 SNPs in 34.8 hours, while MIG^{++} required 2.8 weeks. The parallel version of S-MIG^{++} further reduced this runtime up to 5.1 hours when using 32 CPUs. Very recently, an optimization to the standard likelihood-based $|D'_{i,j}|$ CI estimation was introduced that may further improve the runtime of MIG^{++} and S-MIG^{++} [53].

The performed haplotype blocks recognition over the entire human

genome using the thresholds-free MIG^{++} algorithm showed that LD-based haplotype blocks can span more than 500 Kbp and extend over several millions of base pairs. Such large haplotype blocks are not detected when exploring small regions separately or limiting long range LD between SNPs (e.g., with sliding window approach) to improve runtime.

Recently, the MIG^{++} algorithm was adopted by PLINK [53], which is one of the most widely used software applications for genetic association studies worldwide.

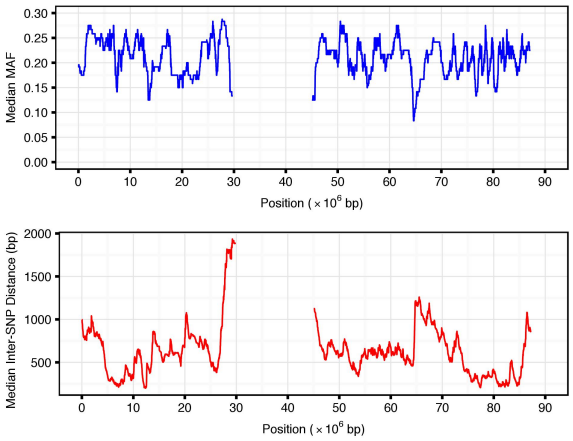
7.2 Future Work

This work was focused only on the runtime and memory efficiency of the first, most computationally expensive, step of Gabriel's method, where all possible haplotype blocks are constructed. In the second step of the algorithm, a greedy approach is applied in order to select a set of non-overlapping blocks that maximize the coverage of the genome. A possible direction of the future research activities is the development and assessment of more sophisticated techniques to ensure the optimality of the block partition.

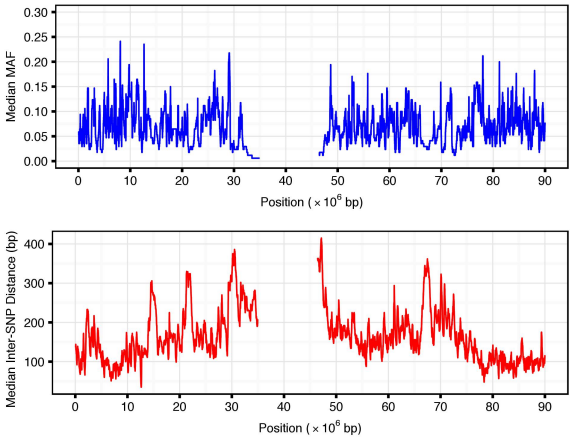
The high runtime and memory efficiency of the developed S-MIG^{++} algorithm allows us to run genome-wide haplotype block partitioning in a reasonably short time. This opens a possibility to apply our introduced sampling-based approach as a preliminary step in other partitioning methods that use more sophisticated haplotype block definitions but are more computationally intensive (Chapter 3). The investigation of such possible applications is the second possible future work direction.

The third possible research direction includes applications of haplotype block partitions in whole-genome association scans such as interactive visualization of LD patterns for interpretation of association results, haplotype association tests, or SNP set-based analyses.

Appendix A



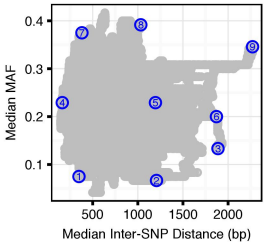
(a) HapMapII



(b) 1000G

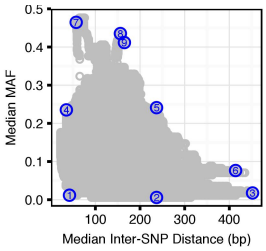
Figure A.1: Median MAF and median inter-SNP distance in sliding regions of 1,000 SNPs on chromosome 16 with no centromere. Similar figures were obtained from all the rest of autosomal chromosomes.

Nr	Genomic Coordinates	Median MAF	Median Inter-SNP Distance (bp)
1	chr8:16110095-16887851	0.0750000	348
2	chr15:40771215-43225829	0.0666667	1207
3	chr16:29576305-34968118	0.1333330	1888
4	chr11:5183617-5559739	0.2291670	163
5	chr20:31767872-33700401	0.2291670	1195
6	chr16:29171648-34806365	0.2000000	1871
7	chr2:15048665-15822444	0.3750000	381
8	chr2:192944318-195085355	0.3916670	1033
9	chr346957741-50137295	0.3458330	2271



(a) HapMapII

Nr	Genomic Coordinates	Median MAF	Median Inter-SNP Distance (bp)
1	chr2:89153688-89307566	0.01176470	42
2	chr16:33617236-34357869	0.00588235	237
3	chr8:48335240-48987674	0.01764710	452
4	chr16:12647838-12700632	0.23529400	35
5	chr13:82569406-82944099	0.24117600	237
6	chr9:66847609-69229598	0.07647060	414
7	chr3:97879829-97981482	0.46470600	57
8	chr13:64621522-64855566	0.43529400	156
9	chr13:64601186-64844532	0.41176500	165



(b) 1000G

Figure A.2: Sampled regions of 1,000 SNPs. The same sampling method was used for regions of 5,000, 10,000, 15,000, 20,000, 25,000 and 30,000 SNPs. Gray points correspond to all generated regions, while blue points correspond to sampled regions.

Bibliography

[1] 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, Bich T. N. Nguyen, Michael C. Norris, John B. Sheehan, Naiping Shen, David Stern, Renee P. Stokowski, Daryl J. Thomas, Mark O. Trulson, Kanan R. Vyas, Kelly A. Frazer, Stephen P. A. Fodor, and David R. Cox. Blocks of Limited Haplotype Diversity Revealed by High-Resolution scanning of Human Chromosome 21. *Science*, 294(5547):1719–1723, 2001.

[2] Kui Zhang, Zhaohui Qin, Ting Chen, Jun S. Liu, Michael S. Waterman, and Fengzhu Sun. HapBlock: haplotype block partitioning and tag SNP selection software using a set of dynamic programming algorithms. *Bioinformatics*, 21(1):131–134, 2005.

[3] Jane Gibson, William Tapper, Sarah Ennis, and Andrew Collins. Exome-based linkage disequilibrium maps of individual genes: functional clustering and relationship to disease. *Human Genetics*, 132(2):233–243, 2013.

[4] David-Alexandre Tregouet, Inke R Konig, Jeanette Erdmann, Alexandru Munteanu, Peter S Braund, Alistair S Hall, Anika Groszhennig, Patrick Linsel-Nitschke, Claire Perret, Maylis DeSuremain, Thomas Meitinger, Ben J Wright, Michael Preuss, Anthony J Balmforth, Stephen G Ball, Christa Meisinger, Cecile Germain, Alun Evans, Dominique Arveiler, Gerald Luc, Jean-Bernard Ruidavets, Caroline Morrison, Pim van der Harst, Stefan Schreiber, Katharina Neureuther, Arne Schafer, Peter Bugert, Nour E El Mokhtari, Jurgen Schrezenmeir, Klaus Stark, Diana Rubin, H-Erich Wichmann, Christian Hengstenberg, Willem Ouwehand, Andreas Ziegler, Laurence Tiret, John R Thompson, Francois Cambien, Heribert Schunkert, and Nilesh J Samani. Genome-wide haplotype association study identifies the SLC22A3-LPAL2-LPA gene cluster as a risk locus for coronary artery disease. *Nature Genetics*, 41(3):283–285, 2009.

[5] J-C Lambert, B Grenier-Boley, D Harold, D Zelenika, V Chouraki, Y Kamatani, K Sleegers, M A Ikram, M Hiltunen, C Reitz, I Mateo, T Feulner, M Bullido, D Galimberti, L Concari, V Alvarez, R Sims, A Gerrish, J Chapman, C Deniz-Naranjo, V Solfrizzi, S Sorbi, B Arosio, G Spalletta, G Siciliano, J Epelbaum, D Hannequin, J-F Dartigues, C Tzourio, C Berr, E M C Schrijvers, R Rogers, G Tosto, F Pasquier, K Bettens, C Van Cauwenberghe, L Fratiglioni, C Graff, M Delepine, R Ferri, C A Reynolds, L Lannfelt, M Ingelsson, J A Prince, C Chillotti, A Pilotto, D Seripa, A Boland, M Mancuso, P Bossu, G Annoni, B Nacmias, P Bosco, F Panza, F Sanchez-Garcia, M Del Zompo, E Coto, M Owen, M O'Donovan, F Valdivieso, P Caffara, E Scarpini, O Combarros, L Buee, D Campion, H Soininen, M Breteler, M Riemenschneider, C Van Broeckhoven, A Alperovitch, M Lathrop, D-A Tregouet, J Williams, and P Amouyel. Genome-wide haplotype association study identifies the FRMD4A gene as a risk locus for Alzheimer's disease. *Molecular Psychiatry*, 18(4):461–470, 2012.

[6] Chi Song, Gary K. Chen, Robert C. Millikan, Christine B. Ambrosone, Esther M. John, Leslie Bernstein, Wei Zheng, Jennifer J. Hu, Regina G. Ziegler, Sarah Nyante, Elisa V. Bandera, Sue A. Ingles, Michael F. Press, Sandra L. Deming, Jorge L. Rodriguez-Gil, Stephen J. Chanock, Peggy Wan, Xin Sheng, Loreall C. Pooler, David J. Van Den Berg, Loic Le Marchand, Laurence N. Kolonel, Brian E. Henderson, Chris A. Haiman, and Daniel O. Stram. A Genome-Wide Scan for Breast Cancer Risk Haplotypes among African American Women. *PLoS ONE*, 8(2):e57298, 2013.

[7] Carmen Dering, Claudia Hemmelmann, Elizabeth Pugh, and Andreas Ziegler. Statistical analysis of rare sequence variants: an overview of collapsing methods. *Genetic Epidemiology*, 35(S1):S12–S17, 2011.

[8] Kai Wang, Mingyao Li, and Maja Bucan. Pathway-Based Approaches for Analysis of Genomewide Association Studies. *American journal of human genetics*, 81(6):1278–1283, 12 2007.

[9] Ashley Petersen, Carolina Alvarez, Scott DeClaire, and Nathan L. Tintle. Assessing Methods for Assigning SNPs to Genes in Gene-Based Tests of Association Using Common Variants. *PLoS ONE*, 8(5):e62161, 05 2013.

[10] Andrea Christoforou, Michael Dondrup, Morten Mattingsdal, Manuel Mattheisen, Sudheer Giddaluru, Markus M. Nöthen, Marcella Rietschel, Sven Cichon, Srdjan Djurovic, Ole A. Andreassen, Inge Jonassen, Vidar M. Steen, Pål Puntervoll, and Stéphanie Le Hellard. Linkage-Disequilibrium-Based Binning Affects the interpretation of GWASs. *The American Journal of Human Genetics*, 90(4):727 – 733, 2012.

[11] Paul Flicek, Ikhlak Ahmed, M. Ridwan Amode, Daniel Barrell, Kathryn Beal, Simon Brent, Denise Carvalho-Silva, Peter Clapham, Guy Coates, Susan Fairley, Stephen Fitzgerald, Laurent Gil, Carlos García-Girón, Leo Gordon, Thibaut Hourlier, Sarah Hunt, Thomas Juettemann, Andreas K. Kähäri, Stephen Keenan, Monika Komorowska, Eugene Kulesha, Ian Longden, Thomas Maurel, William M. McLaren, Matthieu Muffato, Rishi Nag, Bert Overduin, Miguel Pignatelli, Bethan Pritchard, Emily Pritchard, Harpreet Singh Riat, Graham R. S. Ritchie, Magali Ruffier, Michael Schuster, Daniel Sheppard, Daniel Sobral, Kieron Taylor, Anja Thormann, Stephen Trevanion, Simon White, Steven P. Wilder, Bronwen L. Aken, Ewan Birney, Fiona Cunningham, Ian Dunham, Jennifer Harrow, Javier Herrero, Tim J. P. Hubbard, Nathan Johnson, Rhoda Kinsella, Anne Parker, Giulietta Spudich, Andy Yates, Amonida Zadissa, and Stephen M. J. Searle. Ensembl 2013. *Nucleic Acids Research*, 41(D1):D48–D55, 2013.

[12] W. James Kent, Charles W. Sugnet, Terrence S. Furey, Krishna M. Roskin, Tom H. Pringle, Alan M. Zahler, and David Haussler. The Human Genome Browser at UCSC. *Genome Research*, 12(6):996–1006, 2002.

[13] Stacey B. Gabriel, Stephen F. Schaffner, Huy Nguyen, Jamie M. Moore, Jessica Roy, Brendan Blumenstiel, John Higgins, Matthew DeFelice, Amy Lochner, Maura Faggart, Shau Neen Liu-Cordero, Charles Rotimi, Adebowale Adeyemo, Richard Cooper, Ryk Ward, Eric S. Lander, Mark J. Daly, and David Altshuler. The Structure of Haplotype Blocks in the Human Genome. *Science*, 296(5576):2225–2229, 2002.

[14] J. C. Barrett, B. Fry, J. Maller, and M. J. Daly. Haploview: analysis and visualization of LD and haplotype maps. *Bioinformatics*, 21(2):263–265, 2005.

[15] Shaun Purcell, Benjamin Neale, Kathe Todd-Brown, Lori Thomas, Manuel A.R. Ferreira, David Bender, Julian Maller, Pamela Sklar, Paul I.W. de Bakker, Mark J. Daly, and Pak C. Sham. PLINK: A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses. *The American Journal of Human Genetics*, 81(3):559 – 575, 2007.

[16] R. C. Lewontin and Ken-ichi Kojima. The Evolutionary Dynamics of Complex Polymorphisms. *Evolution*, 14(4):458–472, 1960.

[17] R. C. Lewontin. The Interaction of Selection and Linkage. I. General Considerations; Heterotic Models. *Genetics*, 49(1):49–67, 1964.

[18] W.G. Hill and Alan Robertson. Linkage disequilibrium in finite populations. *Theoretical and Applied Genetics*, 38(6):226–231, 1968.

[19] C. 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.

[20] 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.

[21] Carlos Zapata, Gonzalo Alvarez, and Carmen Carollo. Approximate Variance of the Standardized Measure of Gametic Disequilibrium D'. *The American Journal of Human Genetics*, 61(3):771 – 774, 1997.

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

[23] Elisabeth Dawson, Goncalo R Abecasis, Suzannah Bumpstead, Yuan Chen, Sarah Hunt, David M Beare, Jagjit Pabial, Thomas Dibling, Emma Tinsley, Susan Kirby, David Carter, Marianna Papaspyridonos, Simon Livingstone, Rocky Ganske, Elin Lohmussaar, Jana Zernant, Neeme Tonisson, Maido Remm, Reedik Magi, Tarmo Puurand, Jaak Vilo, Ants Kurg, Kate Rice, Panos Deloukas, Richard Mott, Andres Metspalu, David R Bentley, Lon R Cardon, and Ian Dunham. A first-generation linkage disequilibrium map of human chromosome 22. *Nature*, 418(6897):544–548, 2002.

[24] Goncalo R Abecasis, David Altshuler, Adam Auton, Lisa D Brooks, Richard M Durbin, Richard A Gibbs, Matt E Hurles, and Gil A McVean. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–1073, Oct 2010.

[25] 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.

[26] 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.

[27] 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.

[28] Kui Zhang, Fengzhu Sun, Michael S. Waterman, and Ting Chen. Haplotype Block Partition with Limited Resources and Applications to Human Chromosome 21 Haplotype Data. *The American Journal of Human Genetics*, 73(1):63–73, 2003.

[29] Wen-Pei Chen, Che-Lun Hung, and Yaw-Ling Lin. Efficient Haplotype Block Partitioning and Tag SNP Selection Algorithms under Various Constraints. *BioMed Research International*, 2013:13, 2013.

[30] Eric C. Anderson and John Novembre. Finding Haplotype Block Boundaries by Using the Minimum-Description-Length Principle. *The American Journal of Human Genetics*, 73(2):336–354, 2003.

[31] H Mannila, M Koivisto, M Perola, T Varilo, W Hennah, J Ekelund, M Lukk, L Peltonen, and E Ukkonen. Minimum description length block finder, a method to identify haplotype blocks and to compare the strength of block boundaries. *The American Journal of Human Genetics*, 73(1):86–94, 2003.

[32] Gideon Greenspan and Dan Geiger. Model-based inference of haplotype block variation. *Journal of Computational Biology*, 11(2-3):493–504, 2004.

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

[34] Raphaël Mourad, Christine Sinoquet, and Philippe Leray. Probabilistic graphical models for genetic association studies. *Briefings in Bioinformatics*, 13(1):20–33, 2012.

[35] Raphael Mourad, Christine Sinoquet, and Philippe Leray. A hierarchical Bayesian network approach for linkage disequilibrium modeling and data-dimensionality reduction prior to genome-wide association studies. *BMC Bioinformatics*, 12(1):16, 2011.

[36] Russell Schwartz, Bjarni V Halldorsson, Vineet Bafna, Andrew G Clark, and Sorin Istrail. Robustness of inference of haplotype block structure. *Journal of Computational Biology*, 10(1):13–19, 2003.

[37] Thomas G Schulze, Kui Zhang, Yu-Sheng Chen, Nirmala Akula, Fengzhu Sun, and Francis J McMahon. Defining haplotype blocks and tag single-nucleotide polymorphisms in the human genome. *Human Molecular Genetics*, 13(3):335–342, 2004.

[38] Keyue Ding, Kaixin Zhou, Jing Zhang, Joanne Knight, Xuegong Zhang, and Yan Shen. The effect of haplotype-block definitions on

inference of haplotype-block structure and htSNPs selection. *Molecular Biology and Evolution*, 22(1):148–159, 2005.

[39] Amit Indap, Gabor Marth, Craig Struble, Peter Tonellato, and Michael Olivier. Analysis of concordance of different haplotype block partitioning algorithms. *BMC Bioinformatics*, 6(1):303, 2005.

[40] Consortium International HapMap. The International Hapmap Project. *Nature*, 426(6968):789–796, Dec 2003.

[41] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[42] S. Fitzpatrick and A. Scott. Quick simultaneous confidence intervals for multinomial proportions. *Journal of the American Statistical Association*, 1987.

[43] C. P. Quesenberry and D. C. Hurst. Large Sample Simultaneous Confidence Intervals for Multinomial Proportions. *Technometrics*, 6(2):191–195, 1964.

[44] Cristina P. Sison and Joseph Glaz. Simultaneous Confidence Intervals and Sample Size Determination for Multinomial Proportions. *Journal of the American Statistical Association*, (429):366 – 369, 1995.

[45] Zbyněk Šidák. Rectangular Confidence Regions for the Means of Multivariate Normal Distributions. *Journal of the American Statistical Association*, 62(318):626–633, 1967.

[46] Naomi E Allen, Cathie Sudlow, Tim Peakman, and Rory Collins. UK Biobank Data: Come and Get It. *Science Translational Medicine*, 6(224), Feb 2014.

[47] Alkes L. Price, Nick J. Patterson, Robert M. Plenge, Michael E. Weinblatt, Nancy A. Shadick, and David Reich. Principal components analysis corrects for stratification in genome-wide association studies. *Nature Genetics*, 38:904–909, 2006.

[48] Olivier Delaneau, Jonathan Marchini, and Jean-François Zagury. A linear complexity phasing method for thousands of genomes. *Nature Methods*, 9(2):179–181, 2011.

[49] Heejung Shim, Hyonho Chun, Corinne Engelman, and Bret Payseur. Genome-wide association studies using single-nucleotide polymorphisms versus haplotypes: an empirical comparison with data from the North American Rheumatoid Arthritis Consortium. *BMC Proceedings*, 3(Suppl 7):S35, 2009.

[50] Jungsun Park, Junghyun Namkung, Mina Jhun, and Taesung Park. Genome-wide analysis of haplotype interaction for the data from the North American Rheumatoid Arthritis Consortium. *BMC Proceedings*, 3(Suppl 7):S34, 2009.

[51] Aaron J. Lorenz, Martha T. Hamblin, and Jean-Luc Jannink. Performance of Single Nucleotide Polymorphisms versus Haplotypes for Genome-Wide Association Analysis in Barley. *PLoS ONE*, 5(11):e14079, 2010.

[52] Lucia A Hindorff, Jacqueline MacArthur, Joannella Morales, Heather A Junkins, P N Hall, A K Klemm, and Teri A Manolio. A Catalog of Published Genome-Wide Association Studies. www.genome.gov/gwastudies. Accessed December 7, 2013.

[53] Shaun Purcell and Christopher Chang. PLINK 1.9.0. https://www.cog-genomics.org/plink2. Accessed November 20, 2014.