

2007

Parsimony-based genetic algorithm for haplotype resolution and block partitioning

Nadezhda A. Sazonova
West Virginia University

Follow this and additional works at: <https://researchrepository.wvu.edu/etd>

Recommended Citation

Sazonova, Nadezhda A., "Parsimony-based genetic algorithm for haplotype resolution and block partitioning" (2007). *Graduate Theses, Dissertations, and Problem Reports*. 2596.
<https://researchrepository.wvu.edu/etd/2596>

This Dissertation is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Dissertation in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Dissertation has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

Parsimony-based Genetic Algorithm for Haplotype Resolution and Block Partitioning

Nadezhda A. Sazonova

Dissertation submitted to the
College of Engineering and Mineral Resources
At West Virginia University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Computer Science

Approved by

E. James Harner, Ph.D., Co-Chair¹

Elaine M. Eschen, Ph.D., Co-Chair²

Cun-Quan Zhang, Ph.D.³

Robert Mnatsakanov, Ph.D.⁴

Donald A. Adjero, Ph.D.⁵

Arun Ross, Ph.D.⁶

Alan Barnes, Ph.D.

Morgantown, West Virginia

2007

¹Department of Statistics, West Virginia University

²Lane Department of Computer Science and Electrical Engineering, West Virginia University

³Department of Mathematics, West Virginia University

⁴Department of Statistics, West Virginia University

⁵Lane Department of Computer Science and Electrical Engineering, West Virginia University

⁶Lane Department of Computer Science and Electrical Engineering, West Virginia University

KEYWORDS: Single Nucleotide Polymorphism, Haplotyping,
Haplotype Block Partitioning, Genetic algorithm

Abstract

Parsimony-based Genetic Algorithm for Haplotype Resolution and Block Partitioning

by

Nadezhda A. Sazonova

This dissertation proposes a new algorithm for performing simultaneous haplotype resolution and block partitioning. The algorithm is based on genetic algorithm approach and the parsimonious principle. The multilocus LD measure (Normalized Entropy Difference) is used as a block identification criterion. The proposed algorithm incorporates missing data as a part of the model and allows blocks of arbitrary length. In addition, the algorithm provides scores for the block boundaries which represent measures of strength of the boundaries at specific positions. The performance of the proposed algorithm was validated by running it on several publicly available data sets including the HapMap data and comparing results to those of the existing state-of-the-art algorithms. The results show that the proposed genetic algorithm provides the accuracy of haplotype decomposition within the range of the same indicators shown by the other algorithms. The block structure output by our algorithm in general agrees with the block structure for the same data provided by the other algorithms. Thus, the proposed algorithm can be successfully used for block partitioning and haplotype phasing while providing some new valuable features like scores for block boundaries and fully incorporated treatment of missing data. In addition, the proposed algorithm for haplotyping and block partitioning is used in development of the new clustering algorithm for two-population mixed genotype samples. The proposed clustering algorithm extracts from the given genotype sample two clusters with substantially different block structures and finds haplotype resolution and block partitioning for each cluster.

TABLE OF CONTENTS

Abstract	ii
Table of contents	iii
List of Figures	vi
List of Tables	viii
List of Symbols and Abbreviations	ix
Introduction	1
Chapter 1: Background on the problem of haplotyping and blocks partitioning	3
1.1 SNPs, haplotypes and haplotype blocks	3
1.2 The importance of haplotypes and haplotype blocks in the analysis of complex traits	7
1.3 Haplotype decomposition problem	10
1.4 Computational algorithms for haplotyping and block partitioning	11
1.4.1 Haplotyping methods	11
1.4.1.1 Population based methods of haplotyping	11
1.4.1.2 Pedigree based methods of haplotyping	17
1.4.2 Haplotype block partitioning methods	20
1.4.3 Effective haplotyping and the structure of linkage disequilibrium	26
1.5 Goals of this dissertation	29
Chapter 2: Assumptions and underlying concepts of the proposed method	32
2.1 Pure parsimony	32
2.2 Block identification criteria	32
2.3 Block-extension algorithm	36
Chapter 3: Parsimony-based genetic algorithm	40
3.1 Initialization of haplotype decomposition	42
3.2 Obtaining the initial block structure	43
3.3 Assessment of the fitness of haplotype patterns	43
3.4 Selection of the set of fittest patterns within each block	45
3.5 Reconstruction of haplotype patterns within each block	46

3.6 Matching of the pairs of haplotypes in adjacent blocks	47
3.7 Adjustment of the block structure	47
3.8 Evaluation of the current solution	48
3.9 Application of the mutation	49
3.10 Termination of the algorithm	50
3.11 Calculation of scores for the block boundaries	57
Chapter 4: Time complexity of the algorithm	58
4.1 Time complexity of the initialization of the haplotype decomposition	58
4.2 Time complexity of obtaining the initial block structure	58
4.3 Time complexity of the assessment of the fitness of all haplotype patterns within every block	60
4.4 Time complexity of the selection of the fittest subset within every block ...	60
4.5 Time complexity of the construction of the next generation of haplotypes based on the selected haplotype patterns	61
4.6 Time complexity of inter-block transitions	61
4.7 Time complexity of the adjustment of block structure	62
4.8 Time complexity of the evaluation of the current solution	62
4.9 Time complexity of the operation of mutation	63
Chapter 5: Results	65
5.1 Methods for the results evaluation	65
5.1.1 Error rate I	65
5.1.2 Error rate II	65
5.1.3 Switch rate	66
5.2 Results from applying the algorithm to real and simulated data	67
5.2.1 Drysdale data	67
5.2.2 ACE data	69
5.2.3 Daly data	71
5.2.4 Patil data	79
5.2.5 HapMap data	81
5.2.6 Summary of results for HAPLOGEN algorithm	88
Chapter 6: Application of the HAPLOGEN algorithm to population studies	90

6.1 Clustering algorithm: step 1	95
6.2 Clustering algorithm: step 2	96
6.3 Testing the clustering algorithm	97
6.3.1 Simulated ACE data	97
6.3.2 Data from four populations	99
6.3.3 HapMap data	100
6.4 Summary of the clustering algorithm results	103
Chapter 7: Conclusions and future work	105
Bibliography	109
Appendix A. R functions for the results evaluation	115
Appendix B. R code for the Daly data preprocessing	117
Appendix C. R code for the Patil data preprocessing	122
Appendix D. Documentation for <i>haplogen</i> package	125
D.1 Introduction	125
D.2 Getting started	125
D.2.1 Installation for Windows	125
D.2.2 Usage	125
D.2.3 Removal	126
D.3 Input file format	126
D.4 Output	126
D.4.1 Output files for HAPLOGEN algorithm	126
D.4.2 Output files for HAPLOCLUST algorithm	127

LIST OF FIGURES

1.1 SNPs, chromosomes, haplotypes and genotypes	4
1.2 Relationship between Haplotyping problem and the Block partitioning problem ...	26
2.1 Relationship among p-value, number of patterns in a block and the length of a block	35
2.2 Block-extension algorithm represented as a set of binary trees	36
3.1 Haplotype decomposition for the entire set and for an individual genotype within current block partition	41
3.2 Outline of the parsimony-based genetic algorithm <i>HAPLOGEN</i>	42
3.3 Relationship between set of genotypes (X) and currently resolving patterns (Y) in a block	45
3.4 Plots of the global optimization criteria vs. prediction errors	48
3.5 ACE data: relationship between the number of iterations, the global optimization criterion, ds, and the average error rate (as the number of correctly resolved genotypes)	52
3.6 Daly data: relationship between the number of iterations and two global optimization criteria: number of distinct haplotypes, ds; total number of distinct haplotype patterns across all blocks, ncompat	53
3.7 Daly data: relationship between the number of iterations and the two kinds of prediction errors used to compare solutions: average block error and the switch error	54
3.8 Plot of the minimal necessary number of iterations for different data sizes	55
4.1 Block merging process (as a part of a block-extension algorithm) at each level of a corresponding binary tree	59
5.1 Block boundaries scores for the 4 successive runs of the algorithm on the ACE data	70
5.2 Scores for block boundaries from the 4 successive runs of the algorithm	76
5.3 Comparison of the block boundaries for the Daly data from different algorithms: <i>HAPLOGEN</i> , <i>GERBIL</i> , <i>HAP</i> , <i>HaploBlock</i> , original Daly partition	77-78

5.4 Block structure obtained for the CEU data from (a) <i>HAPLOGEN</i> , (b) confidence intervals, (c) four gamete rule and (d) solid spine of LD	85
5.5 Block structure obtained for the YRI data from (a) <i>HAPLOGEN</i> , (b) confidence intervals, (c) four gamete rule and (d) solid spine of LD.....	86
5.6 Block structure obtained for the JPT+CHB data from (a) <i>HAPLOGEN</i> , (b) confidence intervals, (c) four gamete rule and (d) solid spine of LD	87
6.1 Outline of the <i>CLUST</i> algorithm	94
6.2 Scores for block boundaries from (a) the pooled data (YRI+JPT); and the data from each of the clusters inferred by the <i>HAPLOCLUST</i> algorithm: (b) cluster 1, (c) cluster 2	102
6.3 Actual haplotype block partitions obtained for (a) the pooled sample (YRI+JPT); and separately for each of the inferred clusters: (b) cluster 1, (c) cluster 2	103

LIST OF TABLES

5.1 Example 1 for comparison of the error rate II and the switch rate	66
5.2 Example 2 for comparison of the error rate II and the switch rate	67
5.3 Drysdale data and the solutions found by the genetic algorithm	68
5.4 Result of running the algorithm on the ACE data using different optimization criteria	69
5.5 Resolution of the children genotypes of the Daly data at heterogeneous positions into two haplotypes using parents' genotype information	72
5.6 Haplotype patterns within Daly block partition: original and predicted by the algorithm	73
5.7 Typical outcome from running the algorithm for the Daly data	74
5.8 Performance of the proposed genetic algorithm compared to the other 3 algorithms for phasing and block partitioning	75
5.9 Number of shared block boundaries (including close positions) for the Daly data produced by different algorithms	78
5.10 Prediction errors for the missing data	79
5.11 Switch rates for the simulated Patil data	80
5.12 Performance of the <i>HAPLOGEN</i> algorithm on the HapMap phased data on the 500 positions region of chromosome 2 in different population samples	83
5.13 Comparison of the switch rates of several algorithms for haplotype inference for the 500-positions region of chromosome 2 of CEU and YRI data	83
6.1 Results from applying <i>HAPLOCLUST</i> to simulated ACE data I	98
6.2 Results from applying <i>HAPLOCLUST</i> to simulated ACE data II	98
6.3 Results from running <i>HAPLOCLUST</i> on the pairs of samples	100
6.4 Group assignment of genotypes in different pairs of data sets of unrelated individuals	101

LIST OF SYMBOLS AND ABBREVIATIONS

<i>ACE</i>	Angiotensin converting enzyme
<i>app_{jb}</i>	A $(0,1)$ -vectors indicating potential applicability of pattern h_{jb} to every genotype within block b
<i>B</i>	Collection of blocks ($2n \times m$ block matrix) represented by lists of patterns within every block
B	Number of haplotype blocks in a solution
<i>b</i>	Individual block
<i>D'</i>	A measure of linkage disequilibrium between two loci
<i>d(.)</i>	Hamming distance
<i>ds</i>	Number of whole-length distinct haplotypes in a solution
EM	Expectation-Maximization
<i>e_i</i>	Number of errors in the i^{th} genotype's decomposition
<i>err_{II}</i>	Block error rate
<i>f(h_{ib})</i>	Fitness of haplotype pattern h_{ib}
<i>fastPHASE</i>	Algorithm for haplotype resolution
<i>Fb</i>	$2n \times m$ matrix of frequencies of the haplotype patterns within all blocks
<i>G</i>	Genotype data ($n \times m$ genotype matrix)
<i>g</i>	Individual genotype
<i>G_{ib}</i>	Collection of genotypes g compatible within block b with pattern h_{ib}
<i>GAHAP</i>	Genetic algorithm for haplotype resolution
<i>GERBIL</i>	Algorithm for haplotyping and block partitioning
<i>H</i>	Haplotype data (haplotype matrix)
<i>h</i>	Individual haplotype pattern
<i>h_{ib}</i>	Haplotype pattern i within block b
<i>HAP</i>	Algorithm for haplotyping and haplotype block partitioning
<i>HAPAR</i>	Algorithm for haplotype resolution
<i>HAPINFEX</i>	Algorithm for haplotype resolution
<i>HaploBlock</i>	Algorithm for haplotyping and block partitioning

HAPLOGEN	Proposed genetic algorithm for haplotyping and haplotype block partitioning
HAPLOCLUST	Proposed clustering algorithm for haplotyping and haplotype block partitioning in mixed population samples
HAPLOPED	Genetic algorithm for haplotype resolution in pedigree data
HAPLOTYPER	Algorithm for haplotype resolution
HMM	Hidden Markov model
HPM	Haplotype pattern mining
HWE	Hardy-Weinberg equilibrium
I_g	Collection of indices of ambiguous and missing sites in genotype g within block b
l_b	Length of a block b
LD	Linkage disequilibrium
MDL	Minimum description length
MRH	Minimum recombinant haplotyping
n	Number of genotypes in a sample
n_{1j}	Number of genotypes with allele 1 at the j^{th} position
n_{0j}	Number of genotypes with allele 0 at the j^{th} position
n_{2j}	Number of genotypes with ambiguous value at the j^{th} position
n_{compat}	Total number of common haplotype patterns across all blocks
NED	Normalized entropy difference
m	Length of a genotype (SNP sequence)
p_{1j}	Probability of the allele 1 at the j^{th} position of a haplotype
p_i	Frequency of haplotype i
\hat{p}_{1j}	Sample estimate of the probability of allele 1 at the j^{th} position of a haplotype
\tilde{p}_{1j}	Bayesian estimate of the probability of allele 1 at the j^{th} position of a haplotype
$P(h_{ib} G)$	Probability of haplotype pattern h_{ib} within block b given data G
PA	Prior annealing
PedPhase	Algorithm for haplotype resolution in pedigree data

<i>PHASE</i>	Algorithm for haplotype resolution
<i>PGS</i>	Pseudo-Gibbs sampling
<i>PL</i>	Partition legation
<i>PPH</i>	Perfect phylogeny haplotype (problem)
<i>q_i</i>	Expected haplotype equilibrium frequency
<i>R</i>	A statistical computing language
<i>r²</i>	A measure of linkage disequilibrium between two loci
<i>S</i>	Entropy
<i>S_B</i>	Entropy of the observed SNP sequence
<i>S_E</i>	Entropy of the SNP sequence under equilibrium
<i>S_k</i>	Entropy of the <i>kth</i> SNP
<i>SDPHapInfer</i>	Algorithm for haplotype resolution
<i>SR</i>	Switch rate
<i>SNP</i>	Single nucleotide polymorphism
<i>ε</i>	Normalized entropy difference
<i>θ</i>	Recombination fraction

Introduction

Most of the variation in human DNA sequences can be characterized by single nucleotide polymorphisms (SNPs), which are mutations at a single nucleotide position. Variation in the human genome underlies the differentiating features present in the population. Humans are diploid organisms, i.e., each chromosome is made of two distinct copies which are separately called haplotypes. The completion of the Human Genome Project that produced sequenced human DNA brings out a new topic for genomic research: the construction of a full Haplotype Map. Available molecular technologies do not allow cheap and efficient haplotype sequencing (also called haplotyping), which produces the genotype decomposition into the pair of haplotypes. Thus, the problem of haplotyping heavily relies on computational methods. The importance of a full Haplotype Map of the human genome should not be underestimated. It is extremely valuable in the large-scale analysis of complex human diseases, which are represented by combinations of multiple linked mutations and a set of environmental factors. For this reason, haplotype-based analyses have proven to be much more powerful in mapping complex human diseases than single-locus (SNP) based studies. Moreover, recent studies have demonstrated that the human genome has discrete block structures. Considering haplotypes within some particular block facilitates further analysis of complex human diseases.

There have been plenty of methods suggested for the use in either the haplotype decomposition or the block partitioning of the set of genotypes. These two problems, however, are known to be interrelated in the sense that successful genotype phasing depends on the availability of block partitioning and vice versa: block partitioning is mostly possible with the resolved haplotypes. None of the existing algorithms can really perform haplotype decomposition and block partitioning at the same time.

Recently new approaches have been developed that now allow us to simultaneously perform haplotyping and block partitioning while providing good accuracy and speed. The method proposed in this dissertation overcomes most of the deficiencies of the existing methods while providing competitive accuracy and speed. In addition, the proposed algorithm for haplotyping and block partitioning exhibits a new

and useful feature characterizing the block structure in the form of scores for block boundaries. In addition, the proposed algorithm is extended to the two-population case when the genotype sample consists of representative individuals from each population. This extension is designed to separate the two populations with a high degree of accuracy and to find the haplotype resolution and block structure in each group.

Chapter 1

Background on the problem of haplotyping and block partitioning

1.1 SNPs, haplotypes and haplotype blocks

DNA sequences taken from any two individuals are known to be 99.9% identical [1], i.e., mutations present in the human genome account for only about 0.1% of the differences. Single nucleotide polymorphisms (*SNPs*) are the genetic markers that represent the most common type of mutations, i.e., those expressed by changes in a single position within DNA sequence, which are observed in at least 5-10% of the population. Most SNPs are *bi-allelic*, that is, they are defined by only two possible nucleotides (*alleles*) at their specific positions. It was determined [2] that bi-allelic SNPs occur about once in every 600 base pairs in the DNA sequence. The discovery of SNPs has been progressing very rapidly: 2.1 million SNPs were identified by 2001, and by the end of 2003 this number had approached 5.7 million [3]. By the end of April, 2007, the number of SNPs in NCBI dbSNP database was over 11.87 million. The widespread information on SNPs has made them available for extensive research in various fields; in particular, SNPs are found to be extremely useful in identifying the genes related to complex human diseases [1, 4].

Each chromosome is comprised of two copies, each called a *haplotype*. These two haplotypes considered together (or conflated) are called the *genotype* (or *unphased genotype*). The genotype does not have information about which nucleotide base (or allele when referring to a SNP) corresponds to which chromosome (haplotype) out of the two. It can only list the alleles when they are the same on each haplotype and mark positions where they are different. Mendelian Law states that exactly one haplotype is inherited from the father and the other from the mother in the process of reproduction. A pair of haplotypes is a result of genotype decomposition and is considered to be a genotype with known phase (*phased genotype*), i.e., genotypes with alleles assigned to one of the two chromosomes. Thus, the haplotype can be characterized by a sequence of SNP alleles occurring at each particular position.

The relationship between SNPs, chromosomes, haplotypes and genotypes is illustrated by Fig. 1.1 (due to Zhang, *et al.* [5]).

SNP		*					*				*																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
-----	--	---	--	--	--	--	---	--	--	--	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Figure 1.1 SNPs, chromosomes, haplotypes and genotypes

Since there are only two choices of alleles for the bi-allelic SNPs, any haplotype can be encoded as a (0,1)-sequence. In turn, any genotype can be encoded as a (0, 1, 2)-sequence, where positions coded by 0's and 1's are called *homogeneous* (or *homozygous* when they are the same on each chromosome/haplotype) and positions coded by 2 are sites where the two haplotypes carry different alleles, and therefore, are called *heterogeneous* (*heterozygous*) or *ambiguous*. For example, two haplotypes and the corresponding genotypes may have the following encoding:

$h_1 : 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1$ haplotype 1
 $h_2 : 0\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1$ haplotype 2
 $g : 0\ 0\ 2\ 2\ 1\ 0\ 0\ 2\ 1\ 0\ 2\ 2\ 2\ 1\ 1$ genotype

SNPs in close proximity are often correlated, meaning that their specific alleles tend to be inherited together. This results in the fact that there is a limited diversity of haplotypes, i.e., in nature far fewer haplotypes occur than combinatorially possible.

It was recently observed [6] that it is possible to partition human haplotypes into distinct blocks each spanning up to 100 kb, which tend to be inherited together (within which no or few recombinations occur). In contrast, recombinations between blocks are rather common. *Recombination* occurs during *meiosis* (i.e., the cell division process in diploid organisms that involves the fusion of chromosomes), when the parental chromosomes get crossed over to result in a new chromosome that consists of the portions of the two original chromosomes of that parent. In the absence of a

recombination event, the haplotype of a child will be identical to one of the two haplotypes of a parent. The fact that some loci tend to almost never undergo recombination is also related to linkage disequilibrium. *Linkage disequilibrium* (LD) among loci represents a nonrandom association (stochastic dependence) between alleles of SNPs [7]. When alleles at two (or more) distinct loci occur in gametes more frequently than expected (given the known allele), the alleles are said to be in linkage disequilibrium. In other words, linkage disequilibrium indicates the tendency of alleles located close to each other on the same chromosome to be inherited together. The linkage disequilibrium tends to wear off over time since more and more recombinations are taking place between any two particular loci. But there are forces that may preserve old links between alleles and even create the new ones. These forces are known as natural selection and genetic drift.

Natural selection is an old concept, introduced by Darwin, which supports the fact that the only mutations that stay in subsequent generations are those that result in better survival. This results in the fact that unfavorable genes or gene combinations have to eventually become extinct.

Genetic drift is a recently developed concept. It supports the idea that some mutations, which are neutral or not necessarily the best, tend to be preserved in some populations, especially in those of smaller sizes. Genetic drift starts out with the fact that in a given population only a fraction of all possible zygotes become mature adults. This may result in a shift in the frequency of alleles and their combinations. Although this is not particularly common for a large population, small populations may experience a sudden and significant effect.

Genetic drift is represented by the population bottleneck and the founder effects. The *population bottleneck* is an evolutionary event resulting in a significant reduction in size (50% or more) of the original population that leads to the fact that some genetic lineages become extinct. Examples of the population bottleneck include natural disasters like floods, draughts, earthquakes, fires [8]. The *founder effect* occurs when a small group separates from the larger population and has essentially no further contact with it, so that the resulting new population develops very distinct genetic lineages with different frequencies than the original population. According to [8], the founder effect is

represented in the American Indians whose population completely lacks type B blood. Another example is the Amish population in North America. These sects were founded by a very small group of migrants from Europe, and since they never attempted to assimilate with the rest of the North Americans, their gene frequencies remain quite different from the surrounding population.

There is an on-going debate on the unique definition of haplotype blocks. Different authors accept a variety of criteria necessary to detect blocks in a genome. Commonly used block identification criteria include haplotype tagging SNP coverage, recombination rates, modeling of ancestral roots, and also linkage disequilibrium. Despite the differences in the block identification criteria, all studies inherently imply that the haplotype blocks are necessarily characterized by low haplotype diversity, and most studies agree that the existence of blocks is somehow related to the recombination events and linkage disequilibrium.

A number of studies used a *diversity criterion* for block detection. Kimmel and Shamir [9, 10] have identified blocks as segments where a small number of common haplotype patterns (usually no more than 5) represent (cover) a significant fraction of the data (70-90%). The optimal block structure is the one based on the minimal total number of common haplotype patterns. Patil et al. [2] and Zhang et al. [11] used another diversity criterion known as the haplotype tagging SNP coverage: the segment of a SNP sequence is considered to be a block if its common haplotypes, covering a significant fraction of data in the sample, can be distinguished using the minimal number of the SNPs (for example, only 2 out of 20 comprising this particular segment). Such SNPs are called the tagging or representative SNPs. The optimal block partition is the one that has the minimal total number of representative SNPs required to distinguish a pre-specified percentage (coverage) of all haplotypes within each block over the entire SNP sequence under study.

Daly [6] and Wang *et al.* [12] have identified the block boundaries by considering *recombination rates*. The blocks are defined as segments with low or zero recombination rates separated by the spots characterized by high recombination. For example, Daly specifies within block inter-marker recombination rates to vary around 1% or lower,

while inter-marker recombination rates on the block boundaries should exceed 4%. Wang *et al.* use the four-gamete test to model the population historical recombination rates.

Greenspan and Geiger [13] and Koivisto *et al.* [14] have developed *statistical models* to determine optimal haplotype block structures. Greenspan and Geiger used Bayesian Networks to model ancestral roots for identifying haplotype blocks. Koivisto employed the Minimum Description Length (MDL) principle to perform statistical model selection.

Many studies [7, 15, 16] specify haplotype blocks as the *regions of increased LD*. Studies done on the patterns of linkage disequilibrium [17, 18, 19, 20, 21] in the human genome have demonstrated its block-like structure: recombinations tend to be concentrated in so called “hot-spots,” whereas the longer stretches of DNA where markers show increased level of linkage disequilibrium have very low haplotype diversity. The *LD-based* definition of a block is now becoming more popular than other block identification criteria mentioned above. The use of LD in the block detection process can also be justified by the fact that this measure can be interpreted as the degree of the strength of a block, especially if supported by a test of statistical significance.

1.2 The importance of haplotypes and haplotype blocks in the analysis of complex traits

The importance of haplotypes and haplotype blocks is recognized in the application of linkage analysis, association analysis and LD mapping as the analyses of the genetic nature of complex human diseases, the patient-specific response to drugs, and other substances studied by pharmacogenomics and toxicogenomics [4], and also the genetic components of any other particular phenotypic traits [20, 22].

Any phenotype reflected in a genotype is distinguished as Mendelian or complex. A *Mendelian trait* [3, 23] is a trait controlled by one or two genetic loci and has a very clear phenotype associated with it. Such a trait exhibits a simple Mendelian inheritance pattern. As a result, a mutation in a single gene can cause a disease that can be passed to the next generations according to Mendel's laws. Mendelian diseases are usually rare in the population; their examples include [23] sickle-cell anemia, Tay-Sachs disease, cystic fibrosis, xeroderma pigmentosa, etc. Unlike Mendelian traits, *complex traits* are multi-

factorial, and their genetic component is often reflected in several loci; other factors affecting complex traits include environment conditions that serve as a trigger for the expression of such traits [24]. Complex traits have less clear relationships between genotypes and phenotypes due to their multi-factorial nature and are not always inherited in a simple Mendelian fashion. Most human diseases are in fact complex traits, for example [3, 23, 25], arthritis, hypertension, lipid metabolism disorders, certain forms of Alzheimer's disease, cancer, etc. Complex traits that are of interest in medicine include specific reactions to certain drugs and are extensively studied by pharmacogenomics [26, 27]. Naturally, the information on a set of consecutive loci given by haplotypes is of much more value in the analysis of complex traits than that of single loci data.

The search for a disease gene usually starts with linkage analysis. The goal of *linkage analysis* is to find the location of a disease gene(s) relative to some known markers (for example SNPs) [28, 29, 30, 31]. Since linkage implies the tendency of the genes and genetic markers in close proximity to be inherited together, during linkage analysis the recombination rates are measured between the disease gene and the genetic marker with known location. The data on pedigrees are usually used in linkage analysis to determine if recombination has taken place [24, 32]. The presence of a disease gene is only known if it is phenotypically expressed in an individual. If the analysis shows no evidence of recombination (recombination rate is lower than 0.5) between the disease gene and the marker, it is assumed that this gene is located in close proximity to this particular marker. Otherwise, another marker is analyzed the same way. More elaborate analysis involves studying the location of a disease gene relative to several genetic markers (multipoint linkage analysis) [33]. Despite the fact that traditionally genetic markers used in linkage analysis are the SNPs and microsatellites, some studies have shown that the haplotype information may be extremely useful, especially in the case of large, complex pedigrees [34, 35]. In addition, information about haplotype blocks specific to a particular group of genotypes under study is also helpful in the determination of the recombination hot spots (usually located at the boundaries of the blocks).

The main goal of *association analysis* is to identify which particular alleles of a gene are responsible for a given disease [36, 29, 30]. Alleles can be referred to single locus mutations (such as SNPs) or multi-locus sequences represented by haplotypes

within any specific region of interest. Thus, association analysis performs phenotype-genotype association (statistical association between a specific allele and some trait). The association analysis may actually serve as a more powerful alternative to linkage analysis in mapping complex trait loci when using large scale data of unrelated individuals (population based studies) and a dense set of markers [24, 32]. Since haplotypes are more informative than genotypes and SNPs, they provide more effective genetic association analyses [37, 38]. The discovery of haplotype blocks has led to improvements in the studies of complex diseases since having block structure makes it easier to identify the boundaries of the DNA segments of interest [39].

Another type of analysis is called *linkage disequilibrium mapping (LD mapping)*. It uses the information about the location of a region of interest determined through, for example, linkage analysis, and it constructs a plot for a dense set of markers (LD map) within this region to estimate the position of a disease-predisposing mutation using the LD between the markers [40, 38]. LD mapping searches for markers in a region of interest whose alleles are correlated with disease in unrelated individuals. LD mapping is considered to be a more powerful alternative to identifying genes for complex diseases and other genetic traits than linkage analysis. The advantage of LD mapping is that it uses the genetic information of unrelated individuals (as opposed to the pedigrees) [41]. Using this kind of data, LD mapping studies the recombination events traced back thousands of generations (rather than for some particular small family) by estimating the LD between the markers. Another advantage is that LD mapping provides much higher resolution for the regions of interest than linkage analysis. Recent studies [42, 35, 43] have shown that information on haplotypes rather than SNPs increases the effectiveness of LD mapping. There are new linkage disequilibrium mapping methods developed using haplotypes, in particular Haplotype Pattern Mining (HPM) [44, 45], that show very promising results in complex disease mapping and in the discovery of several genes simultaneously. The information about haplotype blocks boundaries may be used during LD mapping to localize the particular alleles [43]; on the other hand, the LD map itself may be predictive about the regions of high LD which are considered to be haplotype blocks.

1.3 Haplotype decomposition problem

Certain molecular technologies [2] can perform haplotype sequencing (also called *haplotyping*) to obtain the genotype decomposition into haplotypes, but the cost of such operations is too high to allow their wide use. Most of the time, the information that molecular technologies (such as the locus-specific polymerase chain reaction or PCR [46]) can supply the researcher is SNP discovery and genotype sequencing. The computational methods inferring haplotypes out of the input genotype data offer attractive alternatives for performing successful haplotyping in terms of the labor, time expenses and monetary cost [47, 48].

The haplotyping problem (also called the *haplotype inference* or *haplotype decomposition* problem) can then be described [49, 50] as follows: given a set of n genotype vectors (sequences of 0, 1 or 2's), produce a set of at most $2n$ distinct haplotype vectors, so that each genotype is associated with exactly 2 haplotypes. Decomposition of genotypes into haplotypes is considered to be valid if each heterogeneous (ambiguous) site decomposes into (0, 1) in a respective pair of haplotypes, and if homogeneous sites (labeled 0 / 1) resolve into (0,0) / (1,1), correspondingly, if certain assumptions are met. Restrictions imposed by assumptions are necessarily a part of the problem since the overall number of feasible solutions for any particular genotype is exponential (2^{k-1} , where k is the number of ambiguous/heterogeneous sites).

There are several widely used assumptions that are usually taken into consideration while resolving the haplotyping problem. Those that apply to genotype data not involving pedigree links most often include pure parsimony and perfect phylogeny [49, 50, 51]. The *parsimonious* principle (also called the *pure parsimony* assumption) states that the true solution tends to resolve the largest number of genotypes in a given set. This principle can also be translated into the criterion of achieving the minimum number of *distinct* haplotypes used in the solution [51]. This principle is partially supported by the fact that the actual number of haplotypes found in natural populations is considerably smaller than the number of combinatorially possible solutions to the haplotype inference problem. It is very popular also due to its biological grounds provided by the population genetics theory. Behind the second most widely used assumption of *perfect phylogeny* lies the concept of the *coalescent*, i.e., a rooted tree

(which itself is called a *perfect phylogeny tree*) that describes the evolutionary history represented by a set of resolved haplotypes [51, 52]. The coalescent model (or perfect phylogeny model) is justified by some molecular observations and is based on the *no-backward mutation* and the *infinite-site* assumptions. The no-backward mutation implies that the mutation in any particular locus (site) only happened once in history. The infinite-site assumption states that the mutations are so sparse in the evolutionary history that at any given time frame there is only one mutation possible. Strictly speaking, the perfect phylogeny assumption does not apply to the infrequent haplotypes or when there is a possibility of recombination. More precisely, this assumption is considered to be realistic only within haplotype blocks. If the information on the haplotype block structure is not available, the effectiveness of these methods may be significantly decreased.

1.4 Computational algorithms for the haplotyping and block partitioning

Up until recently the methods for haplotyping were separated from the methods of haplotype block partitioning in the sense that long-range haplotypes have to be found prior to the haplotype block partitioning process, and vice versa: haplotype decomposition would often benefit from the knowledge of the block structure for the specific group of genotypes under study. Nevertheless, the advances made in both branches (haplotyping and block partitioning methods) have laid out a foundation for further improvements in the field and, thus, need to be discussed.

1.4.1. Haplotyping methods

Haplotyping methods can be applied to two types [53] of data: population (based on the collection of unrelated genotypes) and pedigree (based on the collection of genotypes related by family links).

1.4.1.1 Population based methods of haplotyping

There are basically two groups of haplotyping methods based on population genomic data: combinatorial and statistical. Most of the time these approaches assume that the input genotype data are given by a single block and there are no recombinations.

The input data consists of n individuals with an m -site genotype for each individual. Thus, this data can be represented as a $n \times m$ matrix with entries from the set $\{0, 1, 2\}$.

One of the very first attempts in resolving the haplotyping problem was made by Clark [54]. This approach is implemented in algorithm called *HAPINFERX*. This method is very intuitive and simple. It is based on the above mentioned parsimonious principle which is aimed at finding the solution that resolves the largest number of genotypes in the input sample. Clark's method uses the initial decomposition of genotypes whenever possible: if a genotype consists of at most one heterozygous site, then this genotype can be resolved without ambiguity into unique haplotypes. Otherwise a genotype is considered ambiguous (if it contains more than one heterozygous site). The main idea of Clark's algorithm is to use available haplotypes to resolve the rest of the genotypes: if there is a valid haplotype which is compatible with some genotype, the other valid haplotype can be obtained by applying this current haplotype to the genotype (that is, separating out a haplotype from the genotype using existing compatible haplotype).

Clark's method uses the following simple rule: once there is an initial collection of valid haplotypes, it can be applied one-by-one to the unresolved genotypes to get compatible haplotypes (if there are any). Then the rule can be applied all over again until all genotypes are resolved, or only unresolvable genotypes are left.

Obviously the simplicity of Clark's algorithm trades off for limited applicability and a relatively high error rate. First of all, it cannot get started if there are no unambiguous genotypes available. Second, it cannot guarantee the resolvability of all the genotypes in the input. Third, the high error rate comes from the fact that haplotypes may be mistakenly inferred if a crossover product of two actual haplotypes is identical to another true haplotype. Moreover, the genotype decomposition may depend on the order in which haplotypes (and genotypes) are processed. Fourth, Clark's method cannot reliably handle a large number of linked SNPs and it is vulnerable to violating Hardy-Weinberg equilibrium⁷ (HWE) despite the fact that it is not explicitly based on this assumption.

⁷ Hardy-Weinberg theorem states that in a large population genotypic frequencies will achieve and remain in a state of equilibrium after one generation of random mating and, thus, genotype frequencies can be computed from the allele frequencies [55].

The next step toward inferring haplotypes from genotypes used a statistical approach developed by Excoffier and Slatkin [56]. They employed the *Expectation-Maximization (EM)* algorithm for the successive calculation of haplotype frequencies. Their method is based on the assumption of the Hardy-Weinberg equilibrium (which affects the form of the likelihood function). In the expectation step the current values of the haplotype frequencies are used to calculate the probability of resolving each genotype (phase is unknown) into different pairs of haplotypes. Haplotype frequencies are computed until convergence is reached. In the final stage, the solution of genotypes is based on the maximum probability haplotype resolution for a particular genotype. Even though this algorithm performs better than the Clark's algorithm, it has a lot of disadvantages. First, the algorithm starts by identifying all possible haplotypes for each specific genotype which is exponential in the number of heterozygous loci. The implementation of the algorithm thus becomes limited due to the need to store estimated haplotype frequencies for every possible haplotype in the sample. This increases the space and time complexity of the algorithm tremendously and leads to the fact that the algorithm cannot handle a large number of linked SNPs even though it seems to perform better for large samples of individuals. Another issue that makes this approach inconvenient is that the estimates typically depend on the starting point and therefore have a possibility of falling into a local maximum and not finding the true solution. Also, even though no a-priori assumption is made regarding the linkage equilibrium of the loci, the EM algorithm is most useful in the presence of linkage disequilibrium (i.e., assumes that the data is given as a single block) since otherwise equilibrium alleles would be randomly assigned to possible haplotypes (this is not necessarily a bad feature but it does create some restrictions).

Statistical approaches were further enhanced by the introduction in 2001 of a new method by Stephens, Smith and Donnelly [57], which is a Bayesian method based on *Pseudo-Gibbs Sampling* and is called *PGS*. Its implementation is known as the *PHASE* algorithm. In addition to the Hardy-Weinberg equilibrium, this method makes an assumption on the underlying coalescent model. This model assumes that all haplotypes can be arranged into a tree (called a phylogenetic tree) as though they have descended from one common ancestor through a series of single-site mutations. The coalescent

model affects the prior expectation involved into the model. In the purpose of overcoming the disadvantages of the EM algorithm (as another most recent predecessor) the authors of the PGS method developed an improved algorithm that allows them to reduce the number of possible haplotypes considered in the process. This feature makes the PGS method practical for large samples and a large number of loci. The accuracy of the PGS algorithm is similar to the EM algorithm with a slight improvement, but the PGS method has wider applicability. Another advantage is that this method is also known to be robust to departures from the HWE in data.

Further improvement of the Bayesian methods of haplotyping is reflected in the paper by Niu, Qin, Xu and Liu [58] in which a new method (implemented as the *HAPLOTYPER* program) is developed using the Gibbs sampling with the addition of the two new techniques: *Partition Legation (PL)* and *Prior Annealing (PA)*. The PL and PA techniques improve both the accuracy and capacity in comparison to the previously discussed methods. In particular prior annealing avoids falling into a local maximum. Due to the use of the PL technique, this method is called the *PL method*. One of the distinctive features of this approach is that the prior distribution of the haplotype frequencies is assumed to be Dirichlet, and no assumptions are imposed on the population evolutionary history. The PL method successfully treats missing data and is quite robust to the departure from HWE. Moreover, it is the fastest among other statistical methods and has the smallest error rate.

Kimmel and Shamir in 2005 further explored solutions to the incomplete perfect phylogeny problem [59] where special attention is paid to missing data and developed the probabilistically based algorithm with an expected polynomial run time. Their algorithm has proven to quickly resolve genotype data with high rates of missing entries.

The common feature for all of the combinatorial algorithms developed so far is that they are more superior to the statistical algorithms in terms of the time complexity. This advantage is hard to estimate exactly (since it is not always possible to estimate the time complexity of an algorithm based on convergence), but algorithm performance in terms of speed was evaluated in practice. The time complexity varies slightly among combinatorial methods, but is always polynomial in the input. This fact allows the use of these algorithms for a large number of individuals and SNPs. Among disadvantages one

can list the fact that their output does not carry any statistical information about the population haplotypes (like frequency). Most of these methods assume the Coalescent model (perfect phylogeny model). In general, the combinatorial methods showed better accuracy than the statistical methods.

The combinatorial methods for haplotype resolution are represented by the following set of algorithms. In 2001 Gusfield [60] developed a linear programming algorithm for haplotype resolution. In this work the author analyses Clark's inference rule and the parsimonious principle it is based on. The problem of maximal resolution (alternative formulation of the parsimonious principle where the set of haplotypes that would resolve maximal number of input genotypes is sought) is expressed by means of the directed graphs. The integer (linear) programming algorithm built for solving the maximal resolution problem is proposed and is proven to work efficiently on the simulated data. Gusfield expanded his research of the integer linear programming approach to the haplotyping problem based on pure parsimony and compared its accuracy to some other algorithms [61]. In this work he shows the proposed linear programming algorithm is able to resolve 80-90% of the genotypes correctly, but its efficiency on average is less than, for example, that of *PHASE* algorithm and, in addition, is highly dependent on the level of recombination in the data: the higher the recombination level the less accurate the solution.

In 2002 Gusfield also developed a good, time-efficient algorithm [62]. Its asymptotic running time is $O(nm\alpha(n,m))$, where n is the number of individuals, m is the number of SNPs and $\alpha(n,m)$ is the inverse ackerman function which is a very slowly increasing function and, thus, for all practical purposes can be treated as a constant. The algorithm is based on graphic matroid theory and perfect phylogeny. The algorithm efficiently finds one permitted solution and then in linear time determines if this solution is unique; otherwise, it also finds in linear time the implicit representation of all permitted solutions so that one could easily infer any particular solution in linear time. Although theoretically this algorithm is very efficient, it is very complicated and is not easy to implement. Moreover, there is no information available regarding the real data test results like accuracy.

In an attempt to find a simpler algorithm than that of Gusfield to solve the *PPH* (*Perfect Phylogeny Haplotype*) problem, Eskin, Halperin and Karp developed a new algorithm [63] with asymptotic time $O(nm^2)$. This algorithm also takes a graph-theoretic approach (but different from Gusfield's) and produces a simple linear size data structure which can be used to produce all possible solutions to the problem. Each such solution can be explicitly output in $O(mn)$ time. In addition the authors extend their main algorithm to treat the infrequent haplotypes that do not exactly follow the perfect phylogeny model. The algorithm achieves very low error rate (possibly the lowest in the entire group of haplotyping algorithms), but assumes that the data is represented as a single block. In this first version of their algorithm, the authors do not attempt to incorporate block partitioning into their method.

Another approach to the PPH problem was independently developed by Bafna *et al.* [64] by applying a graph-theoretic approach representing the problem in terms of connected components. The algorithm is simpler to implement than Gusfield's algorithm [62] and has the same time complexity as the algorithm by Eskin, Halperin and Karp, i.e., $O(nm^2)$. It determines whether there is a solution to the PPH problem and, similar to all the above combinatorial algorithms, produces a linear-space data structure to represent all of the solutions. The authors do not present any information regarding the real or simulated data testing results (like the error rate). This makes it hard to compare it to other methods.

Another computational approach was developed by Wang and Xu [65]. They use the parsimonious principle implemented as the greedy branch-and-bound algorithm called *HAPAR*. Their heuristic algorithm makes wide use of the concept of *coverage of a haplotype* (number of genotypes the haplotype can possibly resolve) and achieves accuracy and time complexity comparable to other algorithms in the same group (*PHASE*, *HAPINFERX*, and *HAPLOTYPER*) and has slightly better accuracy for the large samples of data in the presence of the recombinations. The missing data is not incorporated into their model in any way.

Wang, Zhang and Sheng [66] later developed the genetic algorithm *GAHAP* for haplotype resolution based on the parsimonious principle. The algorithm is heuristic in nature and incorporates the cardinality of the solution into the fitness function. The

accuracy provided by *GAHAP* is comparable to that of *HAPAR* (authors' previous work) and has an improved running time for the large data sets (long sequences of SNPs as well as large samples). Like *HAPAR* this algorithm does not treat missing data and is mostly designed to perform haplotyping within known blocks.

Another study was done on the different methods using a pure parsimony approach [67] where Lancia *et al.* proposed several methods. Their exact method is a new integer programming method that uses a polynomial number of variables and constraints. The proposed approximation algorithms are almost linear in the input size.

An approximation algorithm to the optimal haplotype inference problem was also developed later in 2005 by Huang, Chao and Chen [68]. This study was based on maximum parsimony by trying to find the minimum set of haplotypes to resolve the input genotypes. The authors formulate the problem as an integer quadratic problem and propose an iterative semi-definite, programming-based approximation algorithm (*SDPHapInfer* program). The proposed algorithm compares in performance with other haplotyping algorithms like *HAPAR*, *HAPLOTYPER* and *PHASE* and is shown to have comparable error rates and time efficiency with these algorithms.

The overall conclusion on the haplotype inference algorithms is that most performed equally well on short SNP sequences, where there is little possibility of recombinations. The best algorithm from this group, which performs well on long sequences of SNPs, was *PHASE*. For this reason it was selected to perform phasing of the data for the HapMap project [69, 70]. This algorithm, however, requires considerable time.

1.4.1.2 Pedigree based methods of haplotyping

In contrast to population based haplotyping methods, there are also a set of methods that take the pedigree data for the families of related genotypes as input. The known relations between genotypes certainly provide advantages in inferring haplotypes; namely, it is sometimes possible to unambiguously perform genotype resolution into haplotypes. On the other hand, pedigree data are very expensive to obtain and often not available. All of these methods are based on the criterion of the minimum number of recombinants between markers and on the Mendelian Law of inheritance. For this reason,

this set of methods is referred to as *MRH (Minimum Recombinant Haplotyping)* algorithms. It has been proven [71] that the problem of finding a minimum-recombinant haplotype configuration is, in general, NP-hard; but it is possible to develop algorithms with approximate results having good accuracy. Most recent achievements in this area are outlined below.

One of the very first methods to be applied to the genotype pedigree data was proposed by Lin and Speed [72] in 1997. They proposed an algorithm for haplotype decomposition based on a Monte Carlo method. Haplotypes are generated according to the distribution of the joint haplotypes of individuals in a pedigree given their phenotype data. The goal of the algorithm is to find the set of haplotypes with maximum conditional probabilities.

Another approach is described by Qian and Beckmann [71]. Their work represents a six-rule algorithm for the reconstruction of haplotypes in pedigrees. The algorithm does not require the data to satisfy the Hardy-Weinberg equilibrium. The algorithm starts by unambiguously resolving all possible loci according to the Mendelian Law of inheritance. A pedigree of any size is then haplotyped by the sequential and repeated application of a set of rules to each nuclear family (parents-offspring trio) until the successive repetition does not produce any additional results. The algorithm should be performed in both directions: from locus 1 to L , and from locus L to 1, since the results may depend on the direction of the analysis. The time complexity of the algorithm is $O(J^2L^3)$, where J is the size of the family and L is the number of loci. This algorithm was shown to perform very well for small pedigrees but becomes very slow for the data of even moderate sizes.

The next algorithm was developed by Li and Jiang [73, 74]. This is an iterative rule-based algorithm based on blocks of consecutive resolved marker loci (and, thus, is called the *block-extension* algorithm). The authors also present a polynomial time exact algorithm for haplotype reconstruction with zero-recombinant assumption. The algorithm utilizes the system of linear equations over the cyclic group Z_2 and solves it using the method of Gaussian elimination. Similar to the Qian and Beckman's algorithm, the block-extension algorithm starts by unambiguously resolving all possible loci using Mendelian Law of inheritance. Then it uses the fact that the genomic DNA can be

partitioned into long blocks with no or very few recombinants per block. Moreover, the algorithm is also based on the experimental observation that the siblings tend to share haplotype blocks that exist in their parents. The algorithm then uses the longest block in a pedigree to resolve more loci by extending the block. Given any block in the children the algorithm then uses it to resolve loci in parents. Experimental results demonstrated that this algorithm is much more efficient than that of Qian and Beckman since the loci can be resolved faster when considered together in blocks. The authors mention that their algorithm ran less than 1 minute whereas the Qian and Beckman's algorithm required 3 to 4 hours for processing the same data. Theoretical time complexity is $O(dmn)$, where d is the largest number of children in a nuclear family, n is the size of the pedigree, and m is the number of loci. The algorithm was able to recover the true haplotypes in more than 90% of the cases but in general had less accuracy than the Qian and Beckman's due to the exhaustive search capabilities of the later algorithm.

The same authors (Li and Jiang) together with Doi later developed two new dynamic programming algorithms for haplotyping in pedigrees with no mating loops [75]. The first algorithm (*locus-based*) performs dynamic programming on the members of the input pedigree and is efficient when the number of marker loci is bounded by a small constant. The second algorithm (*member-based*) performs dynamic programming on the marker loci and is efficient when the size of the pedigree is small. The key to the effectiveness of both algorithms is that, even though the MRH problem is NP-hard, it is possible to construct a polynomial time algorithm when one of the parameters is bounded by a constant. The time complexity of the first algorithm is $O(nm_02^{3m_0})$, where m_0 is the number of heterozygous loci, and the time complexity of the second algorithm is $O(nm2^{4n})$. It was also shown that, in practice, the locus-based algorithm runs reasonably fast when $m_0 \leq 8$, and the member-based algorithm is efficient when $n \leq 6$. The first algorithm was tested on real and simulated data sets, but no report was provided with regard to its error rate. The computer program called *PedPhase* was created [76] to implement the algorithms [73, 74, 75] proposed by Li, Jiang and Doi described above.

Tapadar, Ghosh and Majumder [77] used a genetic algorithm approach for haplotyping in pedigrees, as implemented in the *HAPLOPED* program. This is a heuristic algorithm that uses an optimization criterion based on the minimum number of

recombinations over possible haplotype resolutions of members of a pedigree. The authors develop a set of elaborate rules for several cases of particular haplotype decompositions. The fitness function for each individual haplotype is constructed in such a way as to express the reverse relationship with the number of recombinants. The algorithm was tested successfully on several sets of data. Limitation of this algorithm include no missing data treatment and the fact that the optimization criteria used is related to the requirement of the high linkage disequilibrium in the data, which is only valid within haplotype blocks. Also, the input requirement for the algorithm is the number of candidate haplotypes N to be considered in each generation. For small N the algorithm runs fast but is not guaranteed to converge to the global minimum; on the other hand, for large N the convergence is guaranteed but the running time increases considerably.

The popularity of the pedigree-based methods of haplotypes is limited due to the rare availability of pedigree data.

1.4.2 Haplotype block partitioning methods

The haplotype block partitioning problem in general is considered to be NP-complete [11], but approximate solutions may have polynomial time complexity. Such approximate solutions were developed with the use of different techniques, e.g., studying the haplotype diversity and a set of representative SNPs [2], investigation of the degree of recombination between pairs of adjacent markers, i.e., searching for the patterns of linkage disequilibrium (LD-based methods) [15], Hidden Markov Models (HMM) [6], dynamic programming [9, 10, 11, 37], and the Minimum Description Length (MDL) method. Also, there is a greedy algorithm that incorporates several block definitions [78]. Almost all of these methods require the input data to be resolved haplotypes and produce haplotype block partitioning with the description of common haplotypes in each block.

One of the first attempts to partition the human genome onto blocks of limited haplotype diversity was made by Patil *et al.* in 2001 [2]. The authors investigated their featured data represented by haploid copies of chromosome 21 isolated in rodent-human somatic cell hybrids (this process made possible to produce whole length haplotypes). The data that they used were large-scale since the length of the SNP sequences was 24,047 SNPs. The block was defined as valid if at least 80% of the input chromosomes

were haplotypes, which were represented more than once in this segment. During the study it was observed that to uniquely identify the haplotype it may only be necessary to consider a very small fraction of SNPs, which are called representative SNPs. In order to find the complete block structure, the authors aimed at minimizing the total number of representative SNPs across all block. To achieve this they used the following greedy optimization algorithm: first, all possible overlapping blocks of length one SNP or larger were considered. Segments that did not satisfy the block definition were excluded from further consideration. Among those remaining overlapping blocks only one was selected with the maximum ratio of total SNPs in the block to the minimal number of SNPs required to uniquely distinguish haplotypes represented more than once in the block (common haplotypes). The rest of the overlapping blocks were discarded. The process then continued until the set of adjacent blocks covering the entire data was obtained. In the sample of 20 chromosomes, a maximum of ten common haplotypes per block were obtained as a result of the algorithm. The algorithm partitions the entire data set of length 24,047 common SNPs into 4135 blocks of SNPs. This study remains a benchmark for the subsequent studies performed on the same data.

The next significant achievement in the area of block partitioning was made by Gabriel *et al.* [15] in 2002. In their study, the authors applied bi-allelic measure of linkage disequilibrium D' to the pairs of markers to investigate the degree of recombination between them. The confidence bounds on D' [22] were studied: a pair of markers was said to be in “strong LD” if the one-sided upper 95% confidence bound on D' was 0.98 and the lower bound was above 0.7. Otherwise, the strong evidence for historical recombination was defined to be present for a pair with the upper confidence bound on D' less than 0.9. The distribution of D' values across the studied regions have revealed the clusters of markers with minimal pairwise evidence of historical recombination. The haplotype blocks were then defined as regions within which only a very small fraction (less than 5%) of pairwise D' values had shown evidence of historical recombination. The authors studied genotype data on samples from four populations: Yoruba, African-American, European and Asian. They determined that, even though their definition of a block was based on recombination, the haplotype blocks revealed, as a result of the study, exhibited very low haplotype diversity (3-5 common haplotypes per

block). The method for haplotype block partitioning proposed by Gabriel *et al.* is distinguished among other methods by the fact that it doesn't require completely resolved haplotypes as an input to produce reliable results (even though it is desirable). However, what is required in this case is sufficiently large samples of genotypes in order to produce dependable values of \mathbf{D}' . The shortcoming of this method is that the use of \mathbf{D}' as a valid measure of linkage disequilibrium for the multi-locus regions is highly arguable, since it may not contain enough information to reveal the long-range patterns of linkage disequilibrium [79].

The HMM (Hidden Markov Model) method developed by Daly *et al.* [6] is based on the idea that every position along a chromosome can be assigned to one of the four ancestral long-range haplotypes. Then the model estimates the maximum-likelihood values (using the EM algorithm) of the historical recombination frequency⁸ (θ) between each pair of markers.

The block structure can then be derived by selecting the boundaries (markers) with a large recombination rate (above 4%). Once the haplotype blocks are defined the subset of SNPs that uniquely distinguish the common (85-90%) haplotypes in each block are determined (although it is not clear how exactly the authors determined the representative SNPs in each block, namely, which additional method they used for this purpose). The obvious weakness of the algorithm is the limit imposed on the maximum number of haplotypes (4) in each block. On the other hand, this may not be a big problem, since empirical studies have shown that four is the typical average number specifying the haplotype block diversity. The advantage of the model is that it calculates the strength of the block boundary (in the form of the recombination rate between end-beginning markers). In addition, the model makes a very realistic assumption about non-zero recombination rates even within blocks (since the more strict assumption of no recombination in blocks may be too strict in reality).

The dynamic programming method [11] developed by Zhang *et al.* incorporates the parsimony principle in most of its variations. It is based on the minimization of the

⁸ Recombination rate between two loci on the same chromosome corresponds to the probability that they end up on different copies of the chromosome; this is the same as the probability that a parent will produce a recombinant (with mixed haplotypes) offspring at a given position [55, 80].

number of representative SNPs (SNPs that can distinguish (cover) 80 to 90% of the haplotypes in a block) within each block, as well as on minimizing the total number of blocks in the partition.

The following notation is assumed: $f(B_i)$ is the minimum number of SNPs required to uniquely distinguish at least α percent (called the coverage) of the unambiguous haplotypes in the i^{th} block B_i . According to the method, the optimal partition is the one minimizing the total number of representative SNPs required to distinguish at least α percent of unambiguous haplotypes in each block for the entire chromosome, $\sum_{i=1}^I f(B_i)$, where I is the number of blocks in the partitioning (unknown in advance). Given K haplotypes and a sequence of n consecutive SNPs, r_i , $i = 1, 2, \dots, n$ is a K -dimensional vector with the k^{th} component $r_i(k) = 0, 1, 2$ being the allele of the k^{th} haplotype at the i^{th} SNP locus: 0 stands for missing data, 1 and 2 represent the two alleles. Thus, a block is defined as r_i, \dots, r_j . Also, two haplotypes are said to be *compatible* if the alleles are the same for the two haplotypes at the loci with no missing data. A haplotype in the block is *ambiguous* if it is compatible with two other haplotypes that are themselves incompatible. Thus, the unambiguous haplotypes can be classified into disjoint groups. All haplotypes in the same group will be treated as identical. The Boolean function $block(r_i, \dots, r_j) = 1$ if at least α percent of the unambiguous haplotypes in the block are represented more than once. This condition should be satisfied in the final partition.

If SN_j is defined as the number of representative SNPs for the optimal block partition of the first j SNPs, r_1, r_2, \dots, r_j and $SN_0 = 0$, then according to the dynamic programming theory,

$$SN_j = \min\{SN_{i-1} + f(r_i, \dots, r_j), \text{ if } 1 \leq i \leq j \text{ and } block(r_i, \dots, r_j) = 1\}. \quad (1.1)$$

It may, in fact, happen that there are several block partitions with the same minimum number of representative SNPs. According to the algorithm, the best partition will be the one with the minimum number of blocks. Let C_i denote the minimum number of blocks of all the block partitions requiring SN_j representative SNPs in the first j SNPs ($C_0 = 0$). Dynamic programming gives the following recursive equation so that the minimum number C_n of blocks in the partition can be computed:

$$C_j = \min\{C_{i-1} + 1, \text{ if } 1 \leq i \leq j \text{ and } \text{block}(r_i, \dots, r_j) = 1 \text{ and } SN_j = SN_{i-1} + f(r_i, \dots, r_j)\} \quad (1.2)$$

The authors also prove that the break points of the blocks follow a Poisson process and show that the overall results are statistically significant. The good feature of this algorithm is that it can be easily adapted to different measures of haplotype quality in a block (like, for example, the algorithm can be based on the haplotype diversity) which depends on the purpose of specific application and, thus, there is a possibility for further improvement. The authors also investigate the influence of the coverage on the block partition and have found that the number of blocks tends to increase with the increase of the coverage. The simplicity of formulation and high effectiveness have made this method highly popular and allowed it to be adapted by subsequent studies [9, 10, 79].

In particular, a dynamic algorithm based on Zhang's was developed by Kimmel *et al.* [9, 10]. It used a different optimization criterion, namely, the minimization of the total number of distinct haplotypes that are observed in all blocks. This algorithm also addresses the problem of treating missing data and is based on a probabilistic model of the haplotype block data. This allows the computation of an optimal score of a block with high probability. The notation of the model is the following: T_i^S , $0 \leq i \leq m$, is the minimum number of blocks in the submatrix of the input matrix induced on the rows of subset s and the columns $1, \dots, i$, where $T_0^S = 0$; for a pair of columns i and j , let B_{ij}^S be the score of the block induced by the rows in s and the columns in $\{i, \dots, j\}$; also P_i , $0 \leq i \leq n$, is defined to be the minimum number of block haplotypes in any row partition of a submatrix induced by columns $\{1, \dots, i\}$. Two dynamic programming equations in this case are:

$$T_i^S = \min_{1 \leq j \leq i-1} T_j^S + B_{ji}^S \quad (1.3)$$

$$\text{and } P_i = \min_{1 \leq j \leq i} P_{j-1} + T_m^{\{j, \dots, i\}}. \quad (1.4)$$

In addition to the highly popular dynamic algorithm in 2003, Zhang proposed a greedy algorithm for haplotype block partitioning called HaploBlockFinder [78]. Zhang's greedy algorithm is flexible to the definition of a block. It actually incorporates a set of block definitions: minimal linkage disequilibrium (using \mathbf{D}'), haplotype coverage, and no historical recombinations. Any of these definitions can be used in the main body of the

algorithm, which is trying to maximize the average number of SNPs within a block. The search is performed in such a way as to find the largest block in the region that satisfies the selected block definition. After at least one such block is found the procedure is repeated for the rest of the subregions. In addition, the algorithm uses tag SNPs to uniquely distinguish common haplotypes. The authors claim, that despite its suboptimal performance, the effectiveness of this algorithm is extremely high when compared to the originally proposed dynamic algorithm. The greedy algorithm is able to achieve optimal solutions in most cases, and it runs about 10 times faster than the dynamic algorithm.

The next method (*Minimum Description Length*) [14] seems to be a significant improvement over the other methods in terms of the quality of the model. The primary advantage of this method lies in the fact that in addition to producing the block partitioning it also finds the probability of a block boundary for each pair of adjacent markers, which provides a measure for evaluating the significance of each block boundary. The MDL method decides among different models on the basis of the minimum of the following function, which is called description length for the data and the model and is expressed by:

$$L(B,D) = L(B) + L(D|B), \quad (1.5)$$

where $L(B)$ is the description of the model and $L(D|B)$ is the description of the data D given the model B .

The authors have tested their method on real and simulated data. The results on the synthetic data show that the method finds the block structure used to generate the data and that the method is very robust against noise. Also the MDL method was applied to the same set of data used in Daly *et al.* (the HMM method), which showed good agreement with those results with the exception of a few minor differences. When noise was added to these data, the block boundaries exhibit very good stability. Another set of the real data was a set of genotypes from the Finland population. In this case, the haplotype blocks did not differ in different Finland subpopulations, which may refer to the presence of a limited set of founder chromosomes shared by all these populations. The overall time complexity of the method is $O(np^3)$ for n observations over p markers.

1.4.3 Effective haplotyping and the structure of linkage disequilibrium

The problems of haplotype inference and block partitioning are both very important and, in addition, very interdependent. From a more general point of view, the effective haplotyping of long SNP sequences largely relies on the knowledge of linkage disequilibrium patterns across the studied genome region since haplotype inference is much easier to model within regions of high LD (or blocks). In turn, the determination of the patterns of LD which is directly related to the block partitioning is more powerful when the phased haplotype information is available. This relationship between the two problems is illustrated in Fig.1.2.

In particular, most of the models developed on block partitioning were using already resolved haplotypes as an input [6, 11, 13, 14]. Also, the models on haplotype inference, which didn't take into account linkage disequilibrium patterns, weren't performing well on the long sequences of SNPs, or worked extremely slowly.

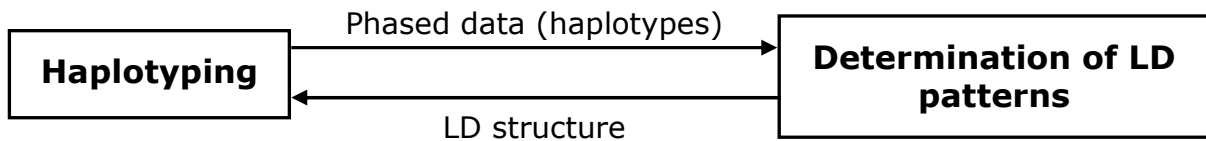


Figure 1.2 Relationship between haplotyping problem and linkage disequilibrium.

Recent studies offered new approaches to haplotyping long SNP sequences, where the models include recombination or LD information suggested by the data. One group of such methods is characterized by implicit use of the LD patterns for haplotype inference. Sheet and Stephens developed a haplotyping algorithm called fastPHASE [81], which uses a Hidden Markov Model to capture patterns of LD across the studied region. The model assumes local clustering of haplotypes into groups. The cluster membership is allowed to be continuously changed over the length of the entire region. Although this algorithm does not produce the patterns of LD, it implicitly incorporates that information. The algorithm performed extremely well when compared to other haplotyping methods and was able to quickly process thousands of genotypes at hundreds of thousands of SNPs. Another algorithm that accounts for linkage disequilibrium was developed by Sun,

Greenwood and Neal [82]. This method is based on a Bayesian Hidden Markov Model and attempts to capture the LD pattern of ancestral haplotypes. It showed very good performance on relatively long SNP sequences compared to other haplotyping methods. Although this algorithm does not actually produce the block structure, it outputs the spacial distribution of recombination hotspots, so it is easy to see where the block boundaries might be.

Another group of methods for effective haplotyping of long SNP sequences is described as “block-based” methods, which are able to simultaneously produce both haplotype resolution and block structure. These methods use fixed block boundaries to perform local haplotype inference. The advantage of these methods is that they also capture the interrelation between the LD to perform the effective haplotype phasing. As the previous group of methods that implicitly use LD information, this group of methods provides improved accuracy and speed in haplotype prediction over the long SNP sequences when compared to the other haplotyping methods. In addition, they explicitly determine the block structure over the studied SNP region.

The literature search for the simultaneous block partitioning and haplotype resolution turned up three main algorithms [83, 84, 85] that allowed such a task to be performed and, in particular, they all provided better accuracy in the haplotype phasing than any of the previous haplotyping approaches. All of these algorithms use the Expectation-Maximization algorithm in some way. The first algorithm, developed by Eskin *et al.* [83] and later implemented as *HAP* software [86], is based on a relaxed version of perfect phylogeny, which is assumed within each block. The authors point out that perfect phylogeny is not a valid assumption for real data sets, even within a single block. The algorithm first finds the block partition from the genotype data using a sliding window of fixed length and then performs the local EM-based haplotype resolution within each such candidate block. The blocks with more than 5 common haplotypes are then discarded. After this process, the haplotype resolution is obtained and the block partitioning is performed again on the resolved haplotypes using a dynamic programming algorithm similar to a previously developed one [11], which is designed to minimize the number of representative SNPs within each block. The missing data are resolved based on the local EM algorithm: the most likely SNP sequence is chosen to replace the missing

data. Even though the algorithm showed results on haplotype resolution better than those from other haplotyping methods [57], there are obvious limitations of this method. First, the number of common haplotypes are restricted to 5, as the most usual bound. Second, the block length is also limited by some maximally allowed length (up to 100 sites), which may contradict real data. As was recently discovered [87, 88], there are recombination “hot” and “cold” spots, i.e., an unevenly distributed linkage disequilibrium across the genome; this indicates that some blocks may in fact be quite long. Third, the main part of the algorithm results in a block partition, together with the list of haplotypes found in each block, from the entire data set and not the resolved full-length haplotypes. To obtain the full-length resolutions for each individual genotype, an additional post-processing step is then undertaken. The tiling of consecutive haplotype block patterns (inter-block transitions) is done using a heuristic that is not a part of the main algorithm. Moreover, as can be seen from the above description, the processes of the haplotyping and block partitioning are not exactly joined into a single model and, thus, are effectively separate methods.

The second algorithm for performing simultaneous block partitioning and haplotyping resolution was developed by Greenspan and Geiger [85], and it resulted in the *HaploBlock* software. The underlying model is based on a Bayesian Network and the MDL (maximum description length) principle to obtain, respectively, the haplotype resolution and the optimal block structure. These two methods are nevertheless joined together into a single model in the sense that the block partitioning parameters are part of the Bayesian Network. Among other advantages is that this method incorporates the inter-block transitions into the main model and no post-processing step is needed to obtain the full-length haplotypes. This algorithm has provided very good results when compared to other haplotyping algorithms [54, 56, 57, 58], and it has also allowed greater variety within each block than the algorithm by Eskin *et al.* described above.

The third algorithm, called *GERBIL*, developed by Kimmel and Shamir [84] is the best so far in terms of speed and accuracy. Similar to the *HaploBlock*, this algorithm is designed to maximize the overall likelihood of the data given the model parameters and also uses the EM approach but, in addition, introduces a new haplotype generating model. The algorithm exhibits simultaneous haplotyping and block partitioning that together lead

to the maximization of the likelihood function. Despite having the best results when compared to those from Eskin *et al.* and the *HaploBlock* algorithm, this method still has some disadvantages. First, the number of common haplotypes within each block is restricted even though rare haplotypes are allowed in the model. Second, the missing data are resolved heuristically and are not the part of the model. Third, similar to the results from Eskin *et al.*, the model produces a sequence of blocks together with their respective haplotype patterns and not full-length haplotypes. The inter-block transitions are performed as an additional step after the main algorithm is completed.

1.5 Goals of this dissertation

In light of the results of the above studies, this dissertation intends to overcome most of the deficiencies emphasized in the previous methods and to provide new valuable features to the solution of haplotyping and block partitioning problems. Specifically, it aims at achieving the following goals: development of a new algorithm that combines haplotype inference and block partitioning in a single model, adequate treatment of missing data, the use of the multi-locus LD measure for block identification, posing no restriction on the size of a block, development of the measure of strength for block boundaries, and also investigation of ways to apply the developed method to finding haplotype resolution and block structures in mixed population samples.

Despite the great advances in efficient haplotyping and, in particular, in the area of simultaneous haplotyping and block partitioning, the current situation still leaves room for further improvement. In particular, most of the existing “block-based” methods of haplotype inference do not achieve truly simultaneous solutions to the two problems of haplotype resolution and block partitioning. These problems are often dealt with separately even though the final solution does result in the resolved haplotypes and the block structure. The first objective of this dissertation is to develop a new, fast, and accurate algorithm that results in obtaining haplotype resolution and the block structure combined in a single model. The method used in this dissertation is a genetic algorithm. The iterative process represented by this kind of algorithm should express the interrelation between the haplotype decompositions and block structure in the sample under study. The outcome of the algorithm should not only include the block structure in

the form of block boundaries and lists of patterns within each block, but also by the set of full-length haplotypes, so that no post-processing of the solution is needed.

One of the biggest problems for algorithms in the area of haplotyping and block partitioning is how to treat missing data. Real genotype data is often incomplete and adequate evaluation of missing data is an extremely valuable attribute. Most of the existing algorithms perform the preprocessing of the missing data as a completely separate operation not related to the model itself. This problem motivates the second objective of this dissertation, that is, to create a model such that the missing data would be part of it and would not require separate treatment. The algorithm has to incorporate missing data by searching for the optimal way to replace them, which would maximize the probability of the solution given the data.

The observed relationship between linkage disequilibrium and the haplotype blocks has made the LD-based definition of the haplotype blocks more popular. Appropriate measures of linkage disequilibrium among loci of a block can be considered as a measure of the strength of the block and, when its value is low, also provides the limited diversity within a block. Existing methods of block partitioning are often based on the alternative definitions of haplotype blocks or, even if they use a LD-based approach, it is not truly multi-locus and, therefore, may not contain enough information on multi-locus patterns. The third objective of the dissertation is to apply the recently developed new multi-locus measure of linkage disequilibrium NED (normalized entropy difference) to the block identification process and investigate the extent of its possible use.

The restriction of the length of a block is often a problem in the block partitioning algorithms. A lot of studies use the 100-position as the upper bound on the length of a block since most of the found blocks fit this limit. Due to the possibility of the existence of longer blocks, no restriction on the block size is always a valuable feature in the block partitioning algorithm. Therefore, the fourth objective of this dissertation is to develop an algorithm that would pose no restriction on the size of a block. The no-restriction-on-block-length requirement would open a valuable possibility to investigate the occurrence of long-range blocks in the real data.

The block structure of a genome is a very complicated concept in the sense that it can hardly be thought of as something rigid. Even though its existence was justified by a number of studies, the block structure always depends on the particular sample of genotypes; and the output produced by different algorithms often varies in the block boundaries. Thus, the exact block structure is very hard to predict. Rather, the block boundaries should be estimated with a probability or score to give an idea of how likely they are to exist at some specific locus. The fifth objective of the dissertation is to provide this kind of measure for the immediate left side of each SNP position.

The haplotype inference and block partitioning problems may become more complicated when dealing with mixed samples from different populations due to possible differences in block structures and the collections of haplotypes among different populations. The sixth goal of the dissertation is to investigate ways to apply the developed algorithm for haplotyping and block partitioning to mixed population samples with unknown population separation in order to infer more adequate haplotype resolutions and block structures.

Chapter 2

Assumptions and underlying concepts of the proposed method

2.1 Pure parsimony

There are two basic assumptions that define the framework of the proposed approach. The first is the principle of *pure parsimony*, which is a widely-used, empirically supported idea. The parsimonious principle aims to minimize the number of distinct haplotypes for resolving the input genotypes. In the proposed genetic algorithm, the parsimonious principle is realized in several aspects. First, the fitness function for each haplotype patterns within any particular block is constructed in such a way as to reflect the possibility of applying the pattern to any genotype from a sample. A greater value of the fitness function would stand, at least most of the time, for the greater number of genotypes, to which this pattern can be applied. Second, the parsimonious principle is expressed in the criteria used to select the best solution (in the form of a haplotype decomposition and block structure): the minimum number of distinct whole-length haplotypes and the minimum total number of common patterns across all blocks.

2.2 Block identification criteria

The second assumption is represented by the operational description of a block, based on the *LD-based block identification criterion*. Here it is assumed that a block is a sequence of genetic markers (SNPs), which exhibits low haplotype diversity and a high LD measure. It should be noted that regions with high LD tend to have a limited haplotype diversity [7]. Haplotype diversity, in turn, is represented by a number of common (covering more than 80% of the data) haplotype patterns in the current block.

There have been several measures used to evaluate the degree of linkage disequilibrium (also called allelic association). The most popular are r^2 and D' [79], which are pair-wise LD measures. They are calculated for two bi-allelic loci, each having alleles 1 and 2, as:

$$D' = \begin{cases} \frac{D}{\min(p_{1\bullet}p_{\bullet 1}, p_{2\bullet}p_{\bullet 2})}, & D > 0 \\ \frac{D}{\min(p_{1\bullet}p_{\bullet 2}, p_{2\bullet}p_{\bullet 1})}, & D < 0 \end{cases} \quad (2.1)$$

$$r^2 = \frac{D^2}{p_{1\bullet}p_{2\bullet}p_{\bullet 1}p_{\bullet 2}}, \text{ where } D = p_{11} - p_{1\bullet}p_{\bullet 1}. \quad (2.2)$$

Here p_{ij} denotes the frequency of haplotype (i,j) and $p_{i\bullet}, p_{\bullet j}$ denote marginal frequencies of alleles i and j at loci 1 and 2, respectively.

The obvious deficiency of these measures is that they are limited to two loci. Several approaches have been developed to amend this deficiency, but most either do not describe multilocus LD directly [89, 90] or are computationally inefficient [91]. Recently there has been one particular LD measure proposed that overcomes even this limitation [7]. It is called the Normalized Entropy Difference ε (NED) [7] and is based on the concept of entropy. A sequence of m bi-allelic loci can be seen as a system with the possible haplotypes as its states. This sequence can assume 2^m states (haplotypes) $(l_1^i, l_2^i, \dots, l_m^i) \in \{0,1\}^m$ of which only t are present. The entropy is used to measure the non-order of the loci sequence [7]:

$$S_B = -\sum_{i=1}^t p_i \log p_i, \text{ where } p_i \text{ denotes the frequency of haplotype } i. \quad (2.3)$$

Under the hypothesis of linkage equilibrium (no stochastic dependence among loci), p_i can be expressed as the product of marginal allele frequencies at all the loci [7]:

$$q_i = q_{a_1^i, \dots, a_m^i} = \prod_{k=1}^m p_{(k)}^{1_{\{a_k^i=0\}}} (1 - p_{(k)})^{1_{\{a_k^i=1\}}}, \quad i \in \{1, \dots, 2^m\}. \quad (2.4)$$

Here a_k^i denotes the allele of the k^{th} SNP position ($k \in \{1, \dots, m\}$) at haplotype i , $p_{(k)}$ denotes the frequency of allele 0 at the k^{th} SNP, and $1_{\{x\}}$ is equal to 1 if condition x is true and 0 otherwise. Then the entropy in the equilibrium case is specified as:

$$S_E = -\sum_{i=1}^{2^m} q_i \log q_i. \quad (2.5)$$

The computational complexity of S_E can be greatly reduced by using a property of the joint entropy of m independent systems. Since in the equilibrium case the joint entropy of m independent loci (systems) can be represented as the sum of m single-system entropies of each locus (SNP), then S_E can simply be calculated as

$$S_E = \sum_{k=1}^m S_k, \quad (2.6)$$

$$\text{where } S_k = -(p_{(k)} \log p_{(k)} + (1 - p_{(k)}) \log(1 - p_{(k)})) \quad (2.7)$$

Thus, S_k is an entropy of the k^{th} SNP.

This way the running time for computing S_E for a single block is reduced from $O(2^m)$ to $O(m)$.

The deviation from the equilibrium state (limited number of present haplotypes and differing frequencies than expected under equilibrium) represents information about the structure of the system. Deviations lead to the decreased value of S_B compared to S_E . The difference

$$\Delta S = S_E - S_B \quad (2.8)$$

is then a measure of the sequence's deviation from the linkage equilibrium state. To allow for comparison between different sets of loci, ΔS is scaled (normalized) by S_E and denoted by ε [7]:

$$\varepsilon = \frac{\Delta S}{S_E} = 1 - \frac{S_B}{S_E}. \quad (2.9)$$

This Normalized Entropy Difference is going to be used throughout the process of block partitioning in order to detect valid blocks.

It is also possible to assess the significance of this LD coefficient. It has been shown [7] that $2n\Delta S \sim \chi^2_{2^m - (m+1)}$ approximately holds, where n is a sample size and m is the length of the SNP sequence (block). This means that the statistic $2n\Delta S$ can be used to test the significance of the deviation of the system from its linkage equilibrium within any particular block.

It should be pointed out that the NED criterion used for block identification has certain limitations. Namely, the asymptotic distribution of its value usually works best for the blocks not exceeding length 8-10 (depending on the number of genotypes in a sample); for longer sequences the NED becomes very insignificant. Thus, it becomes impossible to justify formation of blocks longer than 10 SNPs using the NED criterion. Fig. 2.1 demonstrates this effect in an artificial example where the entropies were calculated using the natural logarithm and were based on the equal frequencies of the two alleles for each site and equal frequencies of the haplotype patterns within block; sample size (number of genotypes) was taken to be 50. The figure illustrates that the p-values

increase after a certain threshold (in this case it is 5) of a block while logic suggests that they continue to decrease. The increase leads to the fact that any block exceeding 8 positions becomes insignificant from the point of view of p-values. To amend this deficiency, another block identification criterion is used in conjunction with the NED measure. This additional criterion is called *coverage of the common patterns*: 4 or 5 patterns with the highest frequencies within every block are assessed on their coverage of the data. The number of the most frequent patterns (4 or 5) is fixed in advance, but as studies show [6, 84] a greater number of common patterns is not practical. The block is considered to be valid for the 80% threshold.

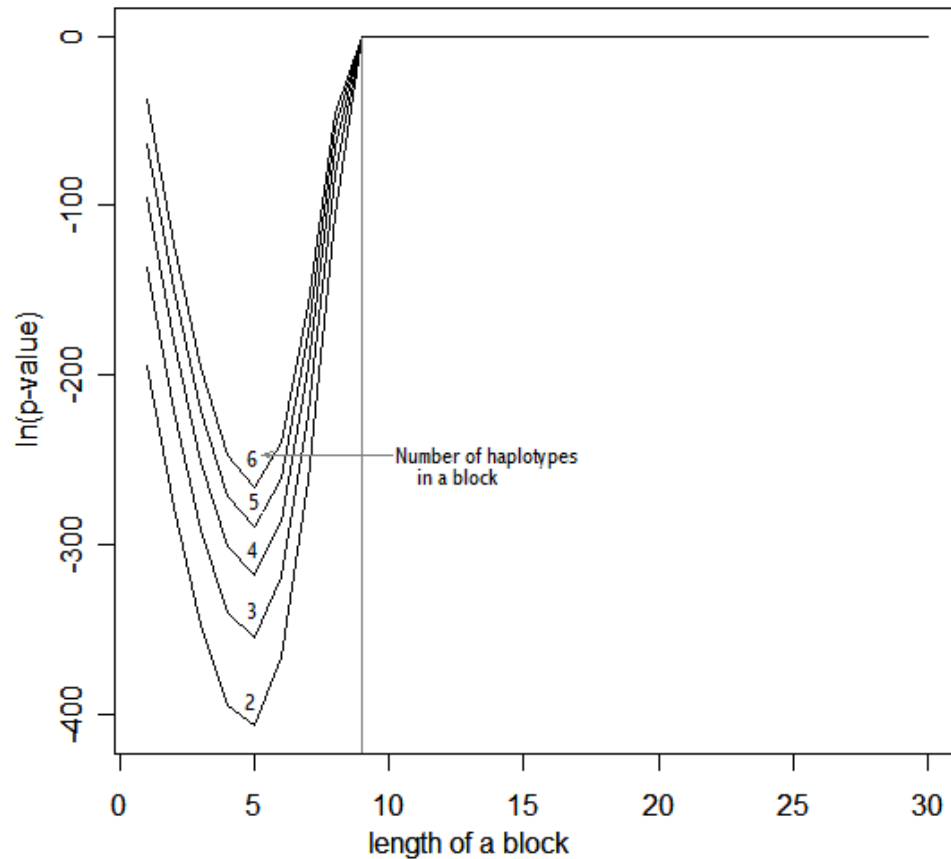


Figure 2.1 Relationship among p-value, number of patterns in a block and the length of a block.

2.3 Block-extension algorithm

The haplotype blocks are produced by a *block-extension* algorithm, which is based on the fact that blocks are usually quite long [6, 15] and is implemented by sequentially combining smaller-size blocks into the bigger ones.

This is a greedy algorithm which incorporates the idea similar to one in the hierarchical clustering: initially every single position (SNP) is considered to be a block; then, at each iteration, pairs of blocks merge into longer blocks if a specific criterion is met. The block-extension algorithm can be used with any meaningful block identification criterion that would allow equivalent comparison for the blocks of different lengths.

The process can be described as follows: some criterion or LD measure (like the Normalized Entropy Difference described above) is calculated and its significance (or threshold value) is assessed for every new block composed of every pair of consecutive blocks. If any newly evaluated block exhibits a sufficient degree of LD, its formation by merging two respective smaller size blocks is justified. Overlapping blocks are also considered, and their strength is compared in order to form the strongest block out of several consecutive overlapping blocks. This idea is implemented as follows: if there are 5 consecutive initial blocks b_1, b_2, b_3, b_4, b_5 , consider, first, the possible new blocks: $b_{12}=b_1 \cup b_2$, $b_{23}=b_2 \cup b_3$. Then select the one with the highest LD. If LD of b_{12} is greater, then block b_{12} is created. If LD of b_{23} wins, compare it with those of b_{34} and create b_{23} only if it still wins; otherwise, if b_{34} wins, compare b_{34} to b_{45} etc.

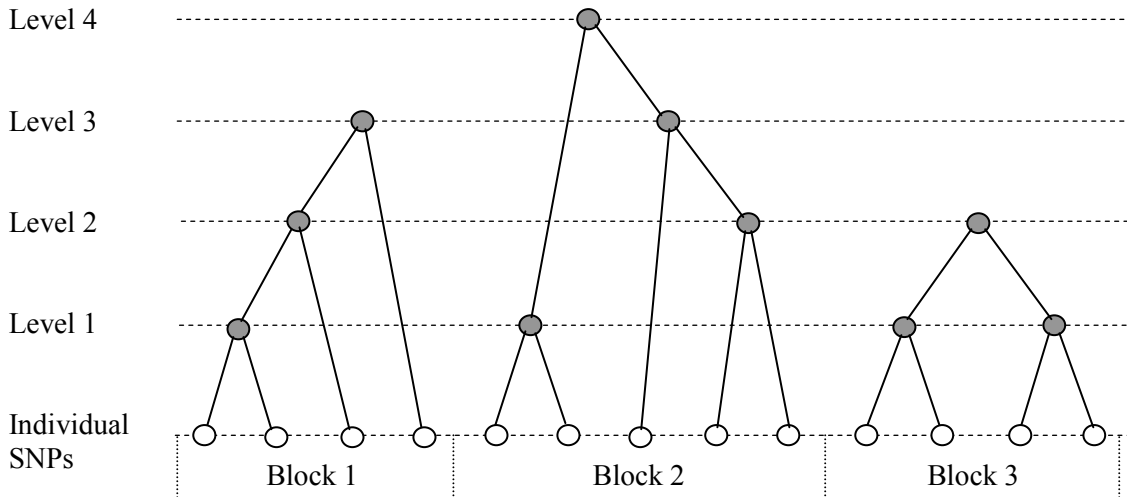


Figure 2.2 Block-extension algorithm represented as a set of binary trees.

When the end of the DNA segment is reached, start over from the beginning and repeat the process with the new set of formed blocks. The entire procedure is repeated until merging is no longer efficient, i.e., the formation of longer blocks does not lead to the significant LD in any of the proposed merges.

Thus, the block-extension algorithm produces a set of binary trees, where each tree represents a particular block and the leaves of each tree are single SNPs (see Fig. 2.2). At each level of a tree, some pairs of the blocks (nodes) from the previous levels are merged into the new upper-level blocks (nodes).

The block-extension algorithm can be proved to solve the following optimization problem under a certain condition:

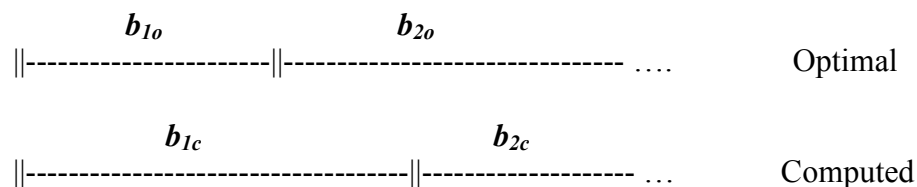
For the given haplotype decomposition find the block structure that maximizes the average linkage disequilibrium LD over all blocks, i.e.

$$\max \frac{\sum_{b=1}^B LB_b}{|B|} \quad (2.10)$$

Where $|B|$ is the number of blocks in the block structure, LD_b is the linkage disequilibrium for the b^{th} block and LD has the following property (*): if S_1 and S_2 are the sequences of consecutive SNP's such that $S_1 \subseteq S_2$, then $LD(S_1) \geq LD(S_2)$.

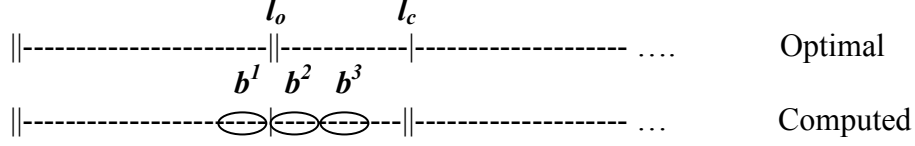
Proof:

Consider the true block partitioning and the block partitioning obtained as a result of the block-extension algorithm. Starting from the left end, find the first non-matching boundary for the two partitionings:



Let the two adjacent blocks at this non-matching block boundary be \mathbf{b}_{1o} and \mathbf{b}_{2o} for the optimal partitioning and \mathbf{b}_{1c} , \mathbf{b}_{2c} for the partitioning computed by the block-extension algorithm. Consider the case when the boundary of the optimal partitioning

between the two blocks l_o occur to the left of the boundary of the computed partitioning l_c .



Since block b_{l_c} is longer than b_{l_o} , at some level during the algorithm block b^1 immediately to the left of the boundary l_o was merged to block b^2 immediately to the right of the same boundary. This merge occurred due to the fact that $LD(b^1 \cup b^2) > LD(b^2 \cup b^3)$.

Now compare the sum of LD's for the 2 blocks from the optimal partitioning (b_{l_o} and b_{2o}) and the sum of LD's for the blocks $b_{l_o} \cup b^2$ (b_{l_o} merged with b^2) and $b_{2o} \setminus b^2$ (block b_{2o} without segment b^2):

$$A_1 = LD(b_{l_o}) + LD(b_{2o}) \quad (2.11)$$

$$A_2 = LD(b_{l_o} \cup b^2) + LD(b_{2o} \setminus b^2) \quad (2.12)$$

Since block b_{2o} can be considered as the union of blocks $b_{2o} \setminus b^2$ and b^2 , due to the property (*) of the LD during this merge, block $b_{2o} \setminus b^2$ has lost some of its LD; denote this loss by c_1 . Then A_1 can also be represented as

$$A_1 = LD(b_{l_o}) + LD(b_{2o}) = LD(b_{l_o}) + LD(b_{2o} \setminus b^2) - c_1. \quad (2.13)$$

Similarly, since during the merge with b^2 block b_{l_o} has lost some degree of LD (denote this loss by c_2), A_2 can be represented by

$$A_2 = LD(b_{l_o} \cup b^2) + LD(b_{2o} \setminus b^2) = LD(b_{l_o}) - c_2 + LD(b_{2o} \setminus b^2). \quad (2.14)$$

Thus, A_1 and A_2 only differ by values of c_1 and c_2 . Since $LD(b^1 \cup b^2) > LD(b^2 \cup b^3)$, b^2 should provide a lower loss of LD when merged with b_{l_o} , than when merged with $b_{2o} \setminus b^2$. Therefore, $c_1 > c_2$ and, thus, $A_1 < A_2$. By that reason the overall objective function (average LD across all blocks) can be improved by replacing b_{l_o} and b_{2o} with $b_{l_o} \cup b^2$ and $b_{2o} \setminus b^2$; thus, the value of the objective function is not optimal. The case when the boundary of the optimal partitioning between the two blocks l_o occur to the right of the boundary of the computed partitioning l_c can be proved similarly. Therefore, the optimal partitioning should coincide with the partitioning computed using the block-extension algorithm. \square

In the version of the block-extension algorithm used in the proposed method for haplotype resolution and block partitioning, two different measures of linkage disequilibrium are used: p-value of the NED and coverage of the most frequent patterns. While the coverage of the common patterns can certainly comply with the property (*) mentioned above in the optimization problem (since extended blocks will tend to provide greater variety of the haplotype pattern), it is not always true for the NED. Since both of these measures are used to find block structure, the algorithm can only guarantee a near-optimal solution to the block partitioning problem at each iteration.

The deficiency of NED described in the previous section is overcome by the use of the second criterion (coverage of the common patterns) in the following way. The block-extension algorithm is applied twice consecutively: first, using the NED (and a significance level of 0.1) and then using the coverage (with the admissible level of 80%). The NED-based algorithm produces blocks that in general do not exceed length 10, and the coverage-based algorithm then uses these smaller-size blocks to create possibly longer blocks, each providing at least 80% coverage. It should also be noted that the coverage criterion cannot be applied by itself since the block-extension algorithm wouldn't work for the trivial block structures (where each SNP stands for a block). It occurs because, for the sequences shorter than 3 positions, the 4 common haplotypes always cover 100% of the data.

Chapter 3

Parsimony-based genetic algorithm

This dissertation proposes the use of a genetic algorithm for haplotype resolution and block partitioning. Previously, genetic algorithms have been applied to the haplotyping problem. As was mentioned earlier, Tapadar, Ghosh and Majumder [77] developed a genetic algorithm for obtaining the haplotype resolution in pedigrees (implemented in *HAPLOPED* program). The fact that this algorithm was designed to be applied only to the pedigree data precludes its widespread use. Pedigree data are very specific, expensive, and, more importantly, not always available for research. Wang, Zhang and Sheng [66] developed a parsimony-based genetic algorithm *GAHAP* for haplotype resolution. Although both *HAPLOPED* and *GAHAP* provided results comparable to similar algorithms at the time, later studies have overridden their results. In particular, none of the previously developed genetic algorithms have simultaneously performed haplotyping and block partitioning, and currently several algorithms successfully perform such a task [83, 84, 85] on independent population data. In this work a new genetic algorithm, *HAPLOGEN*, is proposed. It is able to simultaneously infer haplotype resolution and block structure. In addition, the proposed algorithm is designed to be applied to independent population data that do not require expensive acquisition of the relationship links. The proposed algorithm is developed in form of an R package (with most of the code written in C++) which makes it accessible to any R user and offers the full range of capabilities associated with the R statistical software, including graphical representation of the results.

As with any other genetic algorithm, this method is based on the binary representation of individuals, a fitness function, and operators providing population dynamics (such as mutation). The individuals here are (0,1)-strings representing haplotype patterns within blocks. At each iteration, of the proposed genetic algorithm, there are as many populations as there are blocks since the genetic features of the algorithm are exhibited on the block level. The number of blocks as well as their boundaries vary from iteration to iteration. Within each block the population of individuals is the set of distinct haplotype patterns found in the current haplotype

decomposition. At each iteration every genotype is resolved by two haplotype patterns from each block. Haplotype decomposition for the entire set of genotypes with relation to an individual genotype at any particular iteration is schematically shown in Fig.3.1.

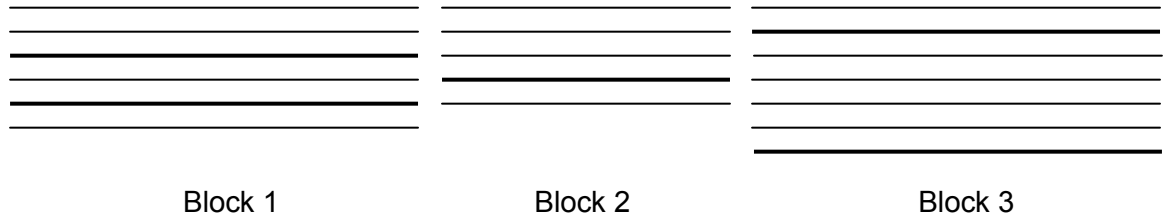


Figure 3.1 Haplotype decomposition for the entire set and for an individual genotype within current block partition (thick lines represent haplotypes used in its resolution).

The proposed genetic algorithm takes the following steps (see Fig. 3.2):

1. Initialize haplotype decomposition.
2. Obtain the initial block structure.
3. Assess the fitness $f(h)$ of each haplotype pattern h within each block.
4. Select the set of fittest patterns within each block according to their fitness function values.
5. Construct the next generation of haplotypes based on the selected haplotype patterns.
6. Perform the inter-block transitions (matching of the pairs of haplotypes).
7. Adjust block structure according to the newly obtained haplotype decomposition.
8. Evaluate the current solution (as a haplotype decomposition and block structure). Exit after the stopping criteria are met.
9. Apply the operation of mutation.

The algorithm is repeated again from step 3 forward until a stopping criterion is met. At each iteration, the current best solution is saved. The last best solution is the final solution to the problem.

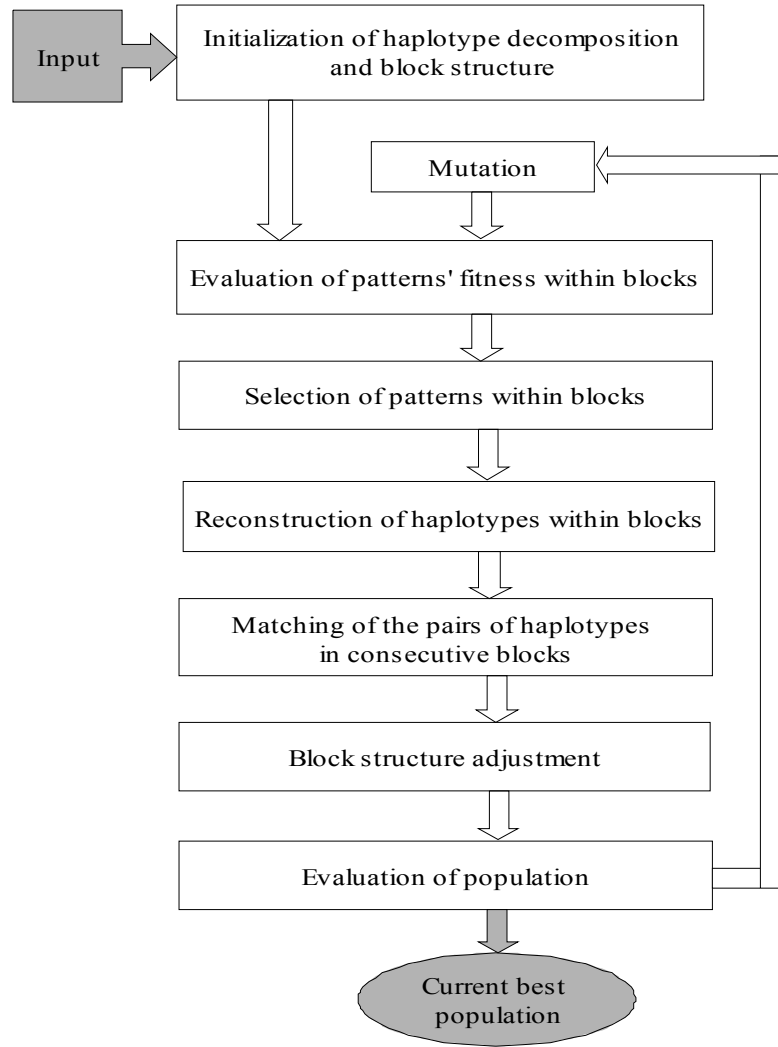


Figure 3.2 Outline of the parsimony-based genetic algorithm *HAPLOGEN*

3.1 Initialization of haplotype decomposition

Each haplotype is represented by a binary string of length m . Initialization is done by randomly obtaining a feasible decomposition for each ambiguous position, i.e., the genotype permutations (0,1) or (1,0). These permutations are randomly assigned with equal probabilities to the pair of haplotypes. Special care is taken at this stage in the sites with *missing information*. Namely, at those positions the initialization is performed randomly so as to assign with equal probabilities one of the four possible values (0,0), (1,1), (1,0) or (0,1) to the pair of haplotypes. Each genotype produces two haplotypes for the population within each block, but only distinct haplotypes are then listed as individuals within every block.

3.2 Obtaining the initial block structure

The current block structure (not only at the initial stage) will be found by applying the *block-extension algorithm* to the trivial block structure, where each SNP position is considered to be a block. As a part of the initialization, a minor modification is performed in each block: all the homogeneous patterns are applied, where possible, to obtain new decompositions of genotypes (original idea is due to Clark [54]). The purpose of this adjustment is to aim the algorithm in the right direction in order to speed up the conversion to the optimal solution.

3.3 Assessment of the fitness of haplotype patterns

The fitness function $f(\mathbf{h})$ of a haplotype pattern \mathbf{h} within a block is represented by the probability of occurrence of any haplotype given the genotype data (within that particular block). The perfect linkage equilibrium between adjacent blocks and random mating is assumed. Within each block, the probability of haplotype pattern \mathbf{h}_{ib} given the genotype data \mathbf{G} can be described as:

$$P(\mathbf{h}_{ib} | \mathbf{G}) = \sum_{\mathbf{g} \in \mathbf{G}_{ib}} \prod_{j \in I_g} p_{1j}^{1_{\{a_j^i=1\}}} (1 - p_{1j})^{1_{\{a_j^i=0\}}}, \quad (3.1)$$

where \mathbf{G}_{ib} is the collection of genotypes \mathbf{g} that are compatible within the current block (i.e., could be used for genotype reconstruction into haplotypes) with pattern \mathbf{h}_{ib} . Then for every such genotype \mathbf{g} , the probability of the pattern \mathbf{h}_{ib} (where I_g is the collection of indices of *ambiguous* and *missing* sites in genotype \mathbf{g} within block \mathbf{b}) is

$$\prod_{j \in I_g} p_{1j}^{1_{\{a_j^i=1\}}} (1 - p_{1j})^{1_{\{a_j^i=0\}}}. \quad (3.2)$$

Here p_{1j} is the probability of the allele 1 in the j^{th} position of haplotype \mathbf{h}_i (respectively, $(1 - p_{1j})$ is the probability of allele 0). It is clear that the greater the number of genotypes compatible with \mathbf{h}_{ib} , the greater the probability of this pattern given the data. For the completely homogeneous genotypes, their contributions to the fitness of corresponding haplotype pattern are doubled. Selection of the patterns with high fitness $f(\mathbf{h}_{ib}) = P(\mathbf{h}_{ib} | \mathbf{G})$ will guarantee the parsimonious principle where the frequencies of haplotypes should be maximized in order to provide a minimum set of distinct haplotypes.

Estimation of the relative frequency p_{1j} of the allele 1 at each site is performed using the following approach. In general, the sample relative frequencies are estimated by the proportion of values at each site that are currently assigned values 1 and 0. The sample estimate of the probability of allele 1, when there is no missing data, is a constant value equal to:

$$\hat{p}_{1j} = \frac{2n_{1j} + n_{2j}}{2n} = \frac{2n_{1j} + n_{2j}}{2(n_{1j} + n_{0j} + n_{2j})}, \quad (3.3)$$

where n_{1j} is the number of genotypes with allele 1 at the j^{th} position, n_{0j} is the number of genotypes with allele 0 at that site, and n_{2j} is the number of genotypes with ambiguous value; n is the total number of genotypes. In the presence of missing data, the sample estimates of probabilities of alleles 1 and 0 will vary from iteration to iteration since some values will currently be assigned to missing data sites.

The lack of information can be corrected via a Bayesian approach by using the Beta distribution as a prior distribution. Current settings allow the use of the likelihood function for the data x (number of successes in n trials) given the parameter p represented by the Binomial probability:

$$f(x | p) = \binom{n}{x} p^x (1 - p)^{n-x}. \quad (3.4)$$

The prior probability density function is selected to be the Beta distribution with parameters $a=3$, $b=1$. Such selection of parameters ($a>b$) leads to the distribution for p which favors values greater than 0.5 and would tend to “draw” posterior estimates toward its expected value, which in this case is given by 0.75 (one of many plausible values). This choice is stipulated by the fact that usually one of the alleles is considered to be rare (and, therefore, is called a *mutation*) while another one is more common. The Beta prior can be represented in general as:

$$g(p) = \frac{1}{B(a,b)} p^{a-1} (1 - p)^{b-1}, \quad (3.5)$$

$$\text{where } 0 \leq p \leq 1 \text{ and } B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \text{ is the Beta function.} \quad (3.6)$$

The posterior density is then represented by the Beta probability density function with new parameters $a^* = a+x$ and $b^* = b+n-x$:

$$g(p|x) = \frac{1}{B(a+x, b+n-x)} p^{a+x-1} (1-p)^{b+n-x-1}. \quad (3.7)$$

Then the Bayesian estimate of the parameter p can be calculated as the expected value from the posterior density:

$$\tilde{p} = E(p|x) = \frac{a^*}{a^* + b^*} = \frac{a+x}{a+b+n}. \quad (3.8)$$

When applied to SNP alleles' frequency estimation, the Bayesian estimate for the probability of allele I becomes:

$$\tilde{p}_{1j} = \frac{a + 2n_{1j} + n_{2j}}{a + b + 2n} = \frac{a + 2n_{1j} + n_{2j}}{a + b + 2(n_{1j} + n_{0j} + n_{2j})}. \quad (3.9)$$

3.4 Selection of the set of fittest patterns within each block

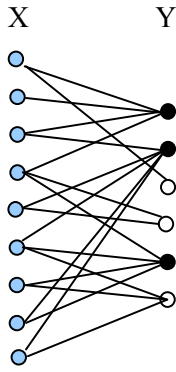


Figure 3.3

Relationship between set of genotypes (X) and currently resolving patterns (Y) in a block.

Selection of the fittest subpopulation within each block is performed in such a way so that every genotype could be potentially resolved (“covered”) by at least one pattern out of the selected ones. Out of all individuals (haplotype patterns) in the current population, a subset is selected randomly without replacement according to their fitness function values. The exact size of this subset is not known in advance, but is determined in the selection process itself as described below.

Selection of patterns within blocks is done proportionally to the partial fitness of a pattern within a block as given by (3.1):

$$P(h_{ib} | G) = \sum_{g \in G_{ib}} \prod_{j \in I_g} p_{1j}^{1_{\{a_j^i=1\}}} (1 - p_{1j})^{1_{\{a_j^i=0\}}}. \quad (3.1)$$

Consider the relationship between genotypes within any particular block and haplotype patterns currently resolving these genotypes. This relationship can be represented as a bipartite graph (X, Y) , where X is the set of genotypes and Y is the set of currently available patterns. Every edge (xy) represents the possibility of resolving genotype x with pattern y . There are at least $2|X|$ edges since every genotype is currently resolved by 2 patterns and can be potentially resolved by some other patterns.

The goal is to find the minimal subset of patterns (vertices of Y) with the highest fitnesses so that there is at least 1 connection (edge) between every vertex of X and this subset of Y . This subset provides coverage for all vertices from X , and it is minimal in the sense that no other subset of this set can fully cover X . Fig. 3.3 shows an example of such a subset of Y as black filled vertices.

The sought after subset of Y is found in the following manner: all genotypes (vertices of X) are marked as “covered” (or potentially covered) by the first selected haplotype pattern; if each next selected pattern does not provide any additional coverage, it is removed from further consideration (and is not selected). Otherwise, new vertices are also marked as “covered” and the pattern is considered to be selected. This process continues until all genotypes are covered by potentially resolving patterns.

3.5 Reconstruction of haplotype patterns within each block

The fittest set of patterns is used to construct the next generation of haplotypes. During this process the genotypes are considered one by one and the following cases may occur:

Case 1: both haplotype patterns currently resolving the genotype under consideration are selected in the fittest subset; there is nothing to be done in this case.

Case 2: at least one of the two haplotypes is not selected. Then choose a randomly selected haplotype to be the “base” where selection is performed proportionally to the fitness from the patterns applicable to this genotype. The pattern is called “applicable” to the genotype i if it can potentially resolve it (but not necessarily in the current decomposition); in other words, if a pattern covers a genotype, then it is applicable to this genotype. After the “base” haplotype is selected, the template for the second haplotype is created. It will be unique if there are no missing data (and no more construction is needed), otherwise, look for the patterns in the fittest set that would apply to the template pattern. If there are several such patterns, randomly select one according to its fitness value. If no pattern fits the template, then identify the missing data in the second haplotype as the exact copy of the same positions in the first haplotype.

The random selection incorporated in the process of the reconstruction of the haplotype patterns is essential in obtaining optimal decomposition (the one providing the

smallest number of patterns within each block) as opposed to the deterministic choice of the applicable pattern with maximal fitness.

3.6 Matching of the pairs of haplotypes in adjacent blocks

Inter-block transitions are made by the choice of the best pairing (tiling) of the 4 haplotype patterns at the block boundary for every genotype. Matching of the pairs of blocks is based on the observed fact that haplotype blocks exhibit long-range dependency [6, 86]. If a certain genotype decomposes into the patterns h_{1a} and h_{1b} in the first block a (in a consecutive pair) and into patterns h_{2a} and h_{2b} in the second block b , then there are only two possible options for the pairing: $\{(h_{1a}, h_{2a}), (h_{1b}, h_{2b})\}$ and $\{(h_{1a}, h_{2b}), (h_{1b}, h_{2a})\}$. The choice is based on the greater of the estimated probabilities of the two options $P((h_{1a}, h_{2a}), (h_{1b}, h_{2b})) = P(h_{1a}, h_{2a}) \cdot P(h_{1b}, h_{2b})$ and $P((h_{1a}, h_{2b}), (h_{1b}, h_{2a})) = P(h_{1a}, h_{2b}) \cdot P(h_{1b}, h_{2a})$. Each of the probabilities $P(h_{ia}, h_{jb}) = P_{ij}$ is estimated by the fraction of genotypes these two consecutive patterns can potentially resolve at the same time. If app_{ia} and app_{jb} are $(0, 1)$ -vectors indicating potential applicability of pattern h_{ia} and h_{jb} to every genotype within respective blocks a and b , then P_{ij} is equal to the scalar product of the two applicability vectors divided by the number of genotypes, i.e., $P_{ij} = (app_{ia}, app_{jb})/n$. If for some genotype its segment within a certain block is entirely homogeneous (leading to the fact that the two probabilities P_{ij} 's are equal), the matching is performed for the two closest non-homogeneous segments (blocks) of this genotype.

3.7 Adjustment of the block structure

After the selection, reconstruction and inter-block transitions are performed, the block structure has to be updated in such a way as to guarantee that the former blocks no longer satisfying the threshold for the block identification are destroyed, and at the same time new blocks are created by using the block-extension algorithm described earlier. The threshold for block destruction is set to be slightly higher (90% coverage) than the one for block creation (80%) since this will allow for the search of more efficient block boundaries, i.e., providing higher average coverage. Thus, only very strong blocks are spared; all others are disassembled into singletons, and the block extension algorithm is then applied.

3.8 Evaluation of the current solution

The new solution in the form of the block structure (defined by the boundaries and the patterns within each block) and the whole-length haplotype decomposition is evaluated according to the parsimonious principle.

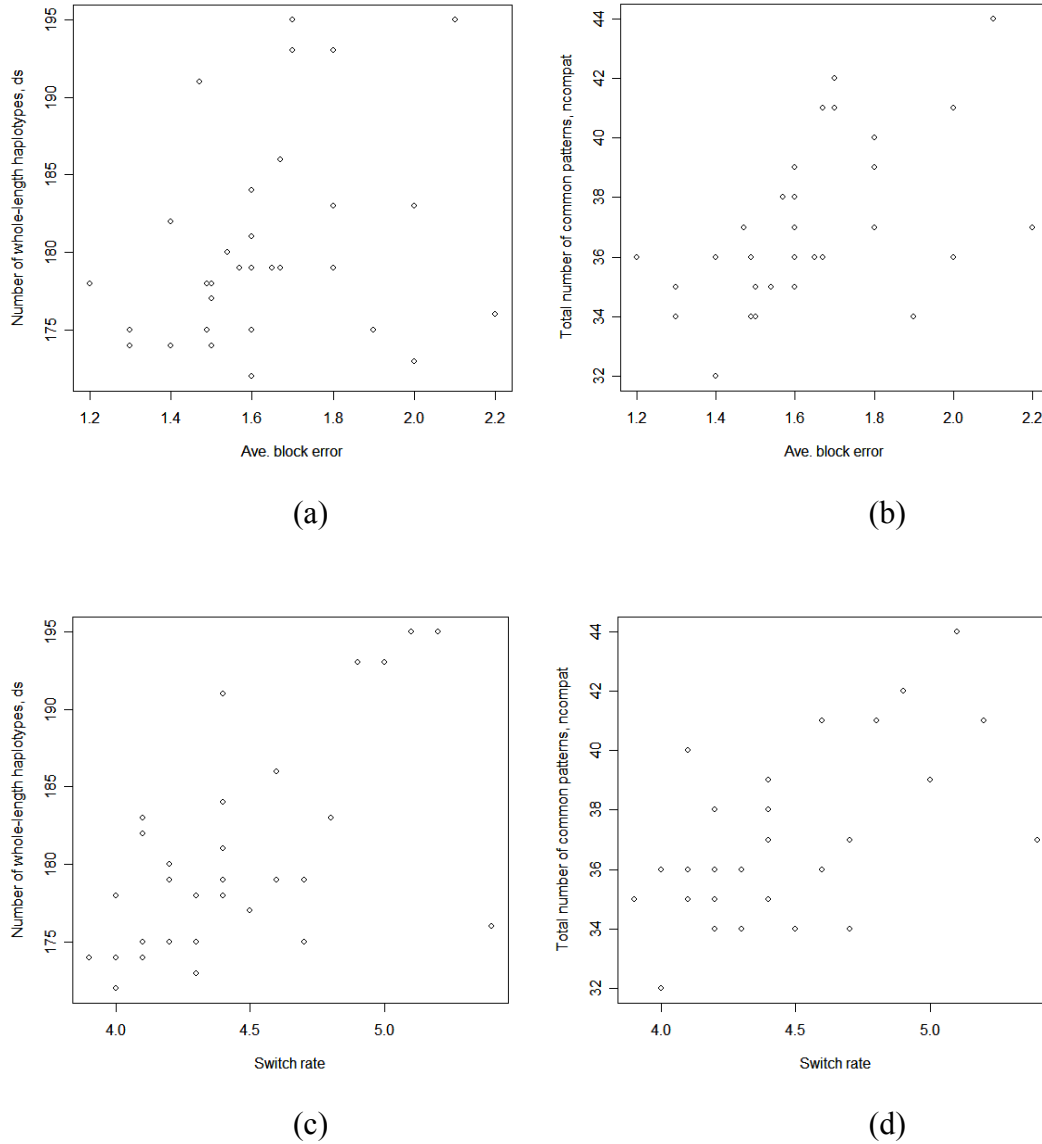


Figure 3.4 Plots of the global optimization criteria vs. prediction errors.

Graphs were constructed using Daly data set illustrating relationships between (a) average block error and the number of whole length haplotypes, *ds*; (b) average block error and the total number of common haplotype patterns (*ncompat*) across all blocks; (c) switch rate and the number of whole-length haplotypes, *ds*; (d) switch rate and the total number of common haplotype patterns (*ncompat*) across all blocks.

On each iteration, the current solution is compared to the current best solution in a form of a double global optimization criterion: the total number of common patterns across all blocks, *ncompat*, and the number of all whole-length haplotypes (common and rare), *ds*. Selection of the best solution is defined by the **minimum** of both of these criteria.

The *ncompat* criterion is used to find the best block structure described by the long blocks with high coverage. The *ds* criterion is needed to predict the best set of the whole-length haplotypes and was used in a number of parsimony-based haplotype decomposition studies [54, 65]. The minimum number of whole-length patterns (*ds*) works especially well with the small size data.

At this step it is also decided whether to continue the genetic search or to terminate the algorithm. The termination of the algorithm is determined by the number of iterations proportional to the size of the input data which is discussed in later sections.

Fig. 3.4 shows how well these two criteria can predict the real data (the Daly data set described below): the accuracy of prediction within each block as measured by the average block error rate is best linearly correlated with the total number of common haplotype patterns, *ncompat*; and the accuracy of prediction of the whole-length haplotypes (measured by the switch rate) is linearly correlated with both of these criteria.

Since minimization of the two criteria (the number of the whole-length haplotypes, *ds*, and the total number of common haplotype patterns, *ncompat*) leads to the minimization of the prediction errors in most cases, their use in the selection of the best solution is justified.

3.9 Application of the mutation

This step can be considered as the beginning of the new iteration since it is used to contribute to the variability of the population(s). The operation of mutation is applied to the pairs of individuals (haplotypes) corresponding to the same genotype and performed only on the heterogeneous sites by switching 0 and 1 with some probability. The missing data sites are not the subject of the mutation operation since they are assumed not to carry any information and are filled in according to the available haplotype patterns. The probability of a mutation is determined by the first order Markov

chain transition probabilities $P(0|1)$, $P(0|0)$, $P(1|1)$ and $P(1|0)$ estimated from the current generation of haplotypes. The use of the first order Markov chain is motivated by the observed existence of dependency between the nucleotides of human DNA that fits first-, second- and higher order Markov chains [92]. In addition, it is reasonable to assume that the Markov chain probabilities estimated by relative frequencies of the adjacent alleles in the current generation of haplotypes may to some extent reflect the frequencies of the longer range haplotype patterns. The computation of the mutation rate is given by the following example:

For the current decomposition $\begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix}$ the probability of the switch (mutation) resulting in $\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}$ is calculated as $p = \frac{P(0|1) + P(1|0)}{P(0|1) + P(1|0) + P(1|1) + P(0|0)}$.

To ensure that good patterns found in previous iterations are not lost during mutation, this operation is only applied to the haplotype pairs with at least one of the two patterns currently covering less than 10% of the resolved haplotypes or when the sample size (the number of genotypes) is very small, i.e., less than 20.

3.10 Termination of the algorithm

There are several stopping (convergence) criteria used in genetic algorithms [93, 94, 95]. Some specify the threshold for the optimization function or its change over time; other algorithms are stopped if there is no improvement in the best fitness value (or the optimization function) over the number of iterations; yet in the others there is simply a bound for the number of iterations. In the case of our genetic algorithm *HAPLOGEN*, the goal of the optimization is to maximize the accuracy of prediction for haplotype decomposition (which cannot be observed); this can only be achieved through controlling some converted optimization criteria that can easily be computed for any given population. The proposed algorithm uses two such global optimization criteria: the number of distinct whole-length haplotypes and the total number of haplotype patterns across all blocks; both are represented by natural numbers. In addition, the proposed algorithm tends to exhibit overtraining (lower accuracy with better global optimization criteria) when applied to the small data sets if the number of iterations is too large, as will be discussed later. This situation hardly favors the use of the convergence criteria based

on the global optimization criteria values. The stopping criterion in the form of the bound on the number of iterations offers a good alternative for that kind of situation. In addition, the bounded number of iterations criterion provides predictable running time for the algorithm given the size for the input data. This feature is extremely valuable in comparing the performance of the proposed algorithm to the existing ones. Due to the reasons described above, the stopping criterion for the *HAPLOGEN* algorithm was chosen to be the bounded number of iterations. This type of termination is used very often [93] and has been proven to show good results. For example, Greenhalgh and Marshall [96] discuss convergence properties for genetic algorithms. They find an upper bound on the number of iterations that would guarantee convergence to a global optimum with a prespecified level of confidence. The overall result of that study is another confirmation that a sufficiently large number of iterations provides good convergence to the optimum.

The number of iterations, sufficient to provide good results for a genetic algorithm, should be proportional to the size of the input data. In case of the studied haplotyping and block partitioning problem, the input data is a genotype matrix; its size is represented by the sample size (the number of rows/genotypes), n , and the number of positions in the SNP sequence (number of columns of the matrix) under study, m . The effect of the size of data on the necessary number of iterations for the *HAPLOGEN* algorithm was analyzed by comparison of the behavior of the global optimization criteria and the prediction error over the different number of iteration. This analysis was performed using two available sets of data: ACE data (described in more detail later) containing 52 positions in 11 genotypes and Daly data containing 103 positions in 129 genotypes.

Analyses of the bounds for the number of iterations was performed by taking subsets of the Daly data and the full ACE data and applying the *HAPLOGEN* algorithm to them. Available measurements of the results from running the algorithm included two observable global optimization criteria (ds, the number of distinct whole-length haplotypes, and ncompat, the total number of common haplotype patterns across all blocks) and prediction errors. Prediction errors are discussed in depth in section 5.1 “Methods for the results evaluation.” Those used here for the stopping criterion analysis included error rate I (proportion of correctly resolved genotypes), error rate II (or block

error rate) and the switch rate. While error rate I is appropriate for short-length SNP sequences, error rate II and the switch rate are more suitable for the long-range data since they both measure the degree of dissimilarity of the computed and the true solution in terms of the number of differing positions.

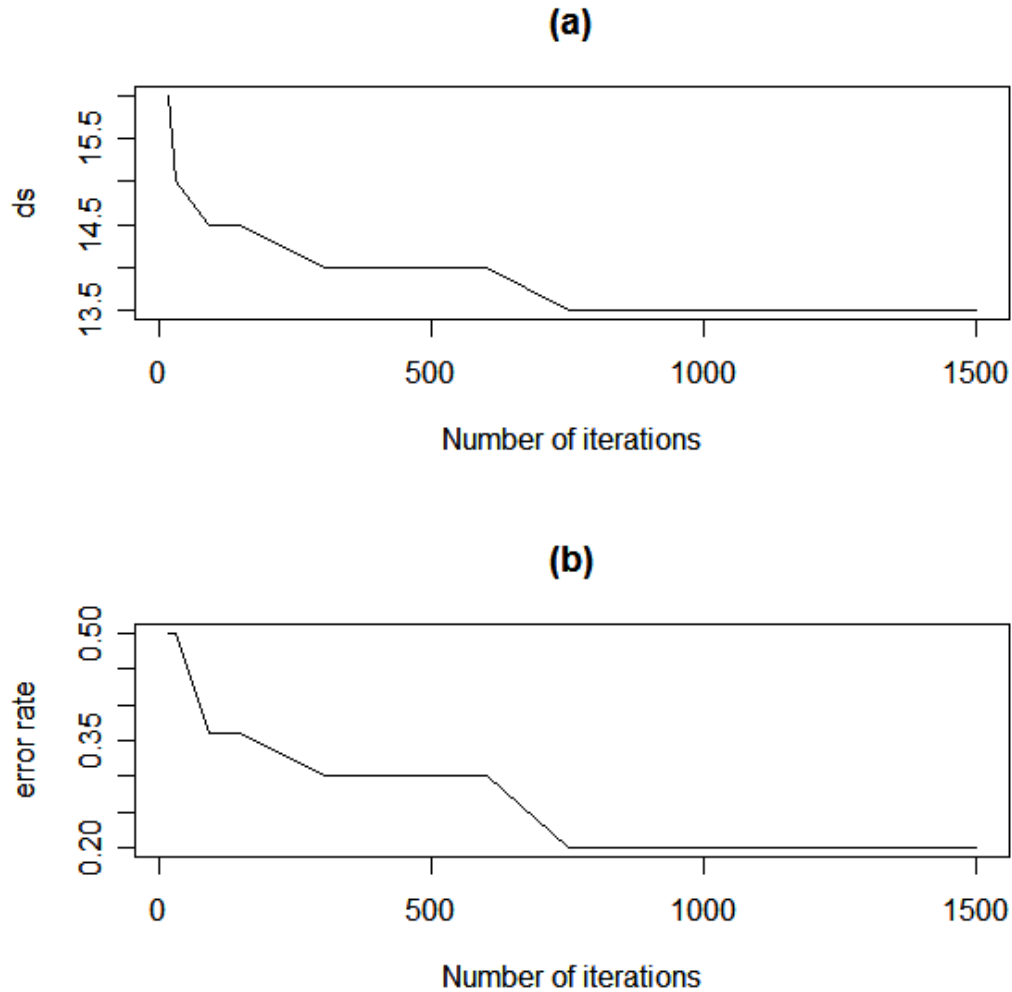


Figure 3.5 ACE data: relationship between the number of iterations and (a) the global optimization criterion, ds , (b) the average error rate (as the number of correctly resolved genotypes).

The relationship between the global optimization criteria and the number of iterations and also between the prediction errors and the number of iterations was analyzed. For each particular set of data, the goal was to determine the saturation point, i.e., the minimal number of iterations sufficient to obtain stable prediction errors or stable global optimization criteria. Due to the random nature of the algorithm results, the values

for all measurements (global optimization criteria and the prediction errors) were obtained as averaged over several (around 10) runs of the algorithm for the same data and the same number of iterations.

The relationship between the number of iterations and the selection criterion, ds , and also between the number of iterations and the error rate I (calculated as the number of correctly resolved genotypes) for the ACE data is given in Fig. 3.5. The figure shows that both indicators stabilize at about 750 iterations.

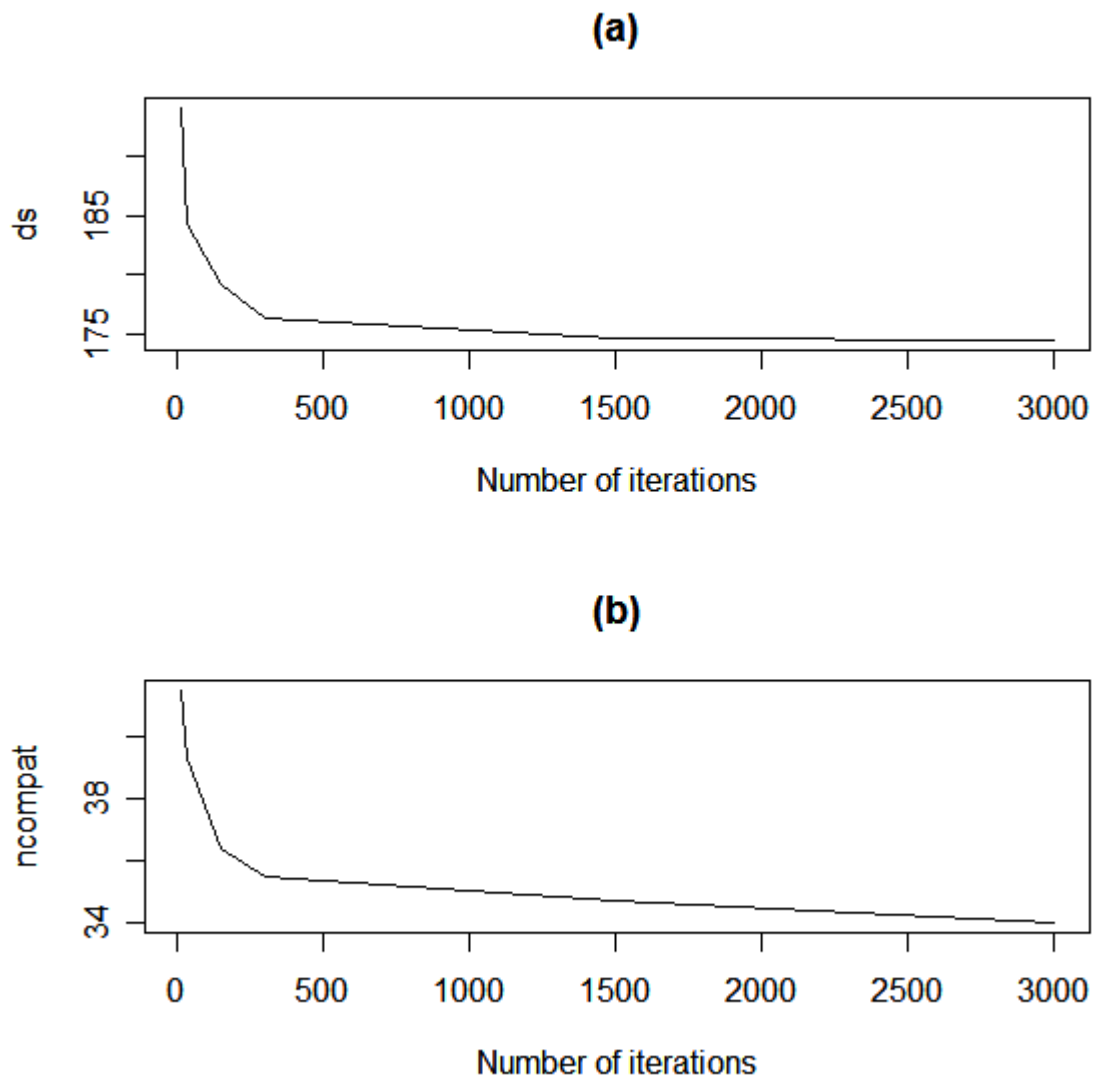


Figure 3.6 Daly data: relationship between the number of iterations and two global optimization criteria: (a) number of distinct haplotypes, ds ; (b) total number of distinct haplotype patterns across all blocks, $ncompat$.

Fig. 3.6 demonstrates the relationship between the two selection criteria (number of distinct haplotypes, *ds*, and the total number of haplotype patterns across all blocks, *ncompat*), for the Daly data. The first criterion, *ds*, becomes almost constant starting from around 1500 iterations while the second criterion continues to fall. The predictive power of the algorithm under different numbers of iterations is shown in Fig. 3.7.

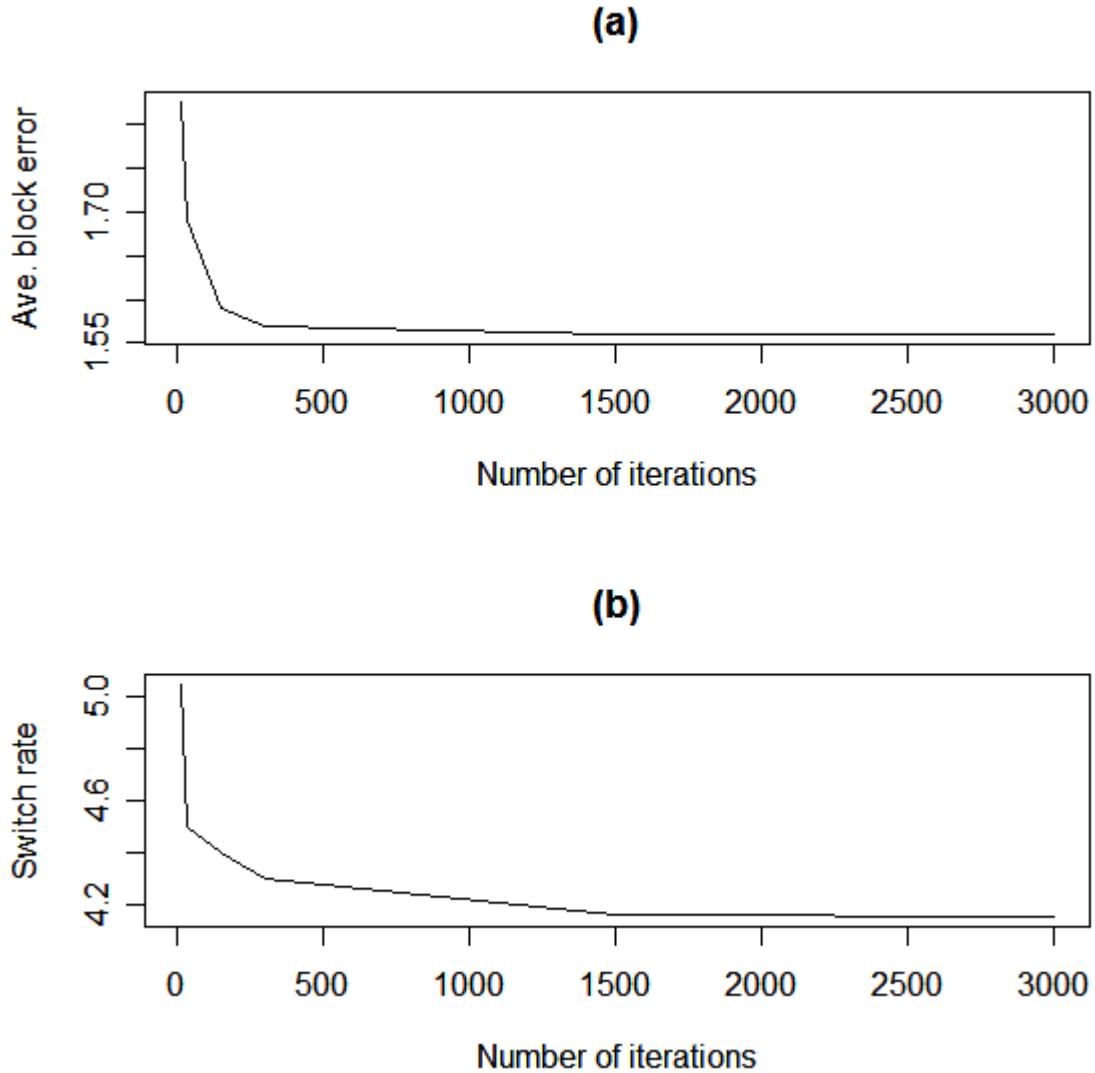


Figure 3.7 Daly data: relationship between the number of iterations and the two kinds of prediction errors used to compare solutions: (a) average block error and (b) the switch error.

Both error rates shown in Fig. 3.7. do not change significantly after about 1500 iterations. Thus, the minimal number of iterations that would guarantee the stabilization of the prediction errors is 1500.

Similar analyses were performed for every partial data obtained by taking subsets of different sizes from the Daly data and by including the full ACE data. For each such subset, the minimal sufficient number of iteration was determined.

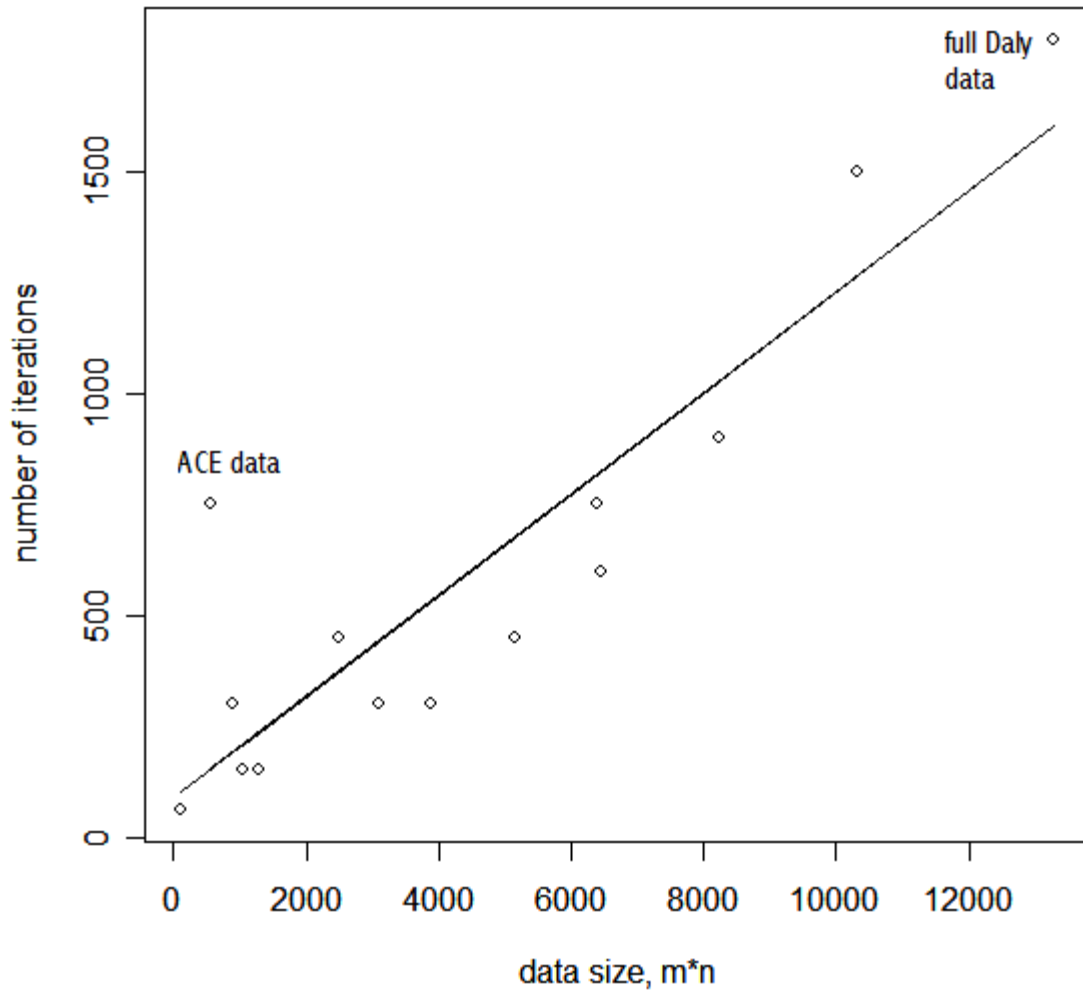


Figure 3.8 Plot of the minimal necessary number of iterations for different data sizes.

Fig. 3.8 represents the plot of the minimal numbers of iterations needed to obtain stabilized (in terms of the prediction errors) solutions. It is reasonable to assume a linear relationship between the data size and the bound for the number of iterations. Therefore,

the linear regression fitted to these data points was constructed resulting in the following equation:

$$it = 89.53 + 0.164(mn). \quad (3.10)$$

Occasionally (as with the ACE data) a larger number of iterations may be needed in order for the algorithm to find correct mutations or correct haplotype resolutions for the majority of genotypes. Since only two data sets were used in the analysis, the incompleteness of information carried by the proposed linear regression can be expected. To amend the situation in the implementation function of the *HAPLOGEN* algorithm, the default bound for the number of iterations (described by the above regression) can be manually replaced by the user within the set of the function's parameters. Since the only observable indicators are the global optimization criteria (*ds* and *ncompat*), the number of iterations may be determined by finding the saturation point for these criteria. Usually, this analysis will provide a larger number of iterations needed to obtain stabilized global optimization criteria, than when the prediction errors are used. In particular, there may be overtraining when a large number of iterations will lead to increases in prediction errors. Although a larger number of iterations doesn't spoil the solution in most cases, this is a very delicate issue due to possible overtraining when the data are short in one dimension, i.e., characterized by a very small sample size (*n* is less than 20) or by very few SNP positions (*m* is less than 20). Overtraining due to a small sample size can be explained by the improper representation of the population. In that case, the algorithm searching for the minimum number of haplotypes may not have enough information to infer common haplotypes correctly and forces incorrect decomposition by haplotypes that are actually rare. On the other hand, when the data are too short (*m* is small) there may not be enough information about the haplotype patterns that are longer than *m*. For example, consider the case where *m* = 5, but the actual length of a block is larger (8). Let there be two genotypes, recorded within the length of 5 SNPs as (0 1 0 0 0) and (0 2 2 0 0) while within a complete block they are (0 1 0 0 0 0 0 0) and (0 2 2 0 0 1 1 1). Then searching for the shortest list of patterns, the algorithm will force the following decomposition:

$$\begin{array}{rcl} 01000 & \rightarrow & 01000 \\ & & 01000 \\ 02200 & \rightarrow & 01000 \\ & & 00100 \end{array}$$

Thus, the following actual (true) decomposition will be missed:

0 1 0 0 0 | 0 0 0 → 0 1 0 0 0 | 0 0 0

0 2 2 0 0 | 1 1 1 → 0 1 1 0 0 | 1 1 1
0 0 0 0 0 | 1 1 1

In general, using very short data for haplotype decomposition or block partitioning is not advised due to the described incompleteness of information and subsequent unreliability of results.

3.11 Calculation of scores for the block boundaries

The scores for the block boundaries at each SNP position are actually a byproduct of the proposed genetic algorithm. Taking advantage of the fact that the algorithm goes through a series of iterations, each score is calculated as a proportion of the time the algorithm selects this position as a block boundary in its current best solution. Each score approximately indicates how likely it is for this particular SNP to have a block boundary immediately to its left. The more some particular boundary appears in a best solution, the stronger this position is considered to be a block boundary.

Chapter 4

Time complexity of the algorithm

The running time of the proposed genetic algorithm depends on the time complexity of each iteration and on the total number of iterations needed to obtain stable results. The necessary number of iterations is discussed in section 3.10 and in general is proportional to the size of the data, i.e., its time complexity is $O(mn)$. The overall time complexity of the algorithm is the product of the time complexity of each iteration and the time complexity of the number of iterations. Running time for each iteration is the sum of the times needed to compete each step.

4.1 Time complexity of the initialization of the haplotype decomposition

Since the initialization process is equivalent to going over the input genotype matrix (of size mn) and filling in the haplotype matrix (of size $2mn$), the time complexity of this step is $O(mn)$. Generation of a random number at each heterogeneous/missing data position is assumed to be constant.

4.2 Time complexity of obtaining the initial block structure

The maximal number of levels in the binary tree, reflecting the block merging process (block-extension algorithm), is $(m-1)$ since this occurs when only one block is created at each level. Also, at each level every pair of adjacent blocks is tested as a potential block. Fig. 4.1 demonstrates this process on a haplotype matrix at some particular level: the rectangles represent current blocks (with vertical dimension reflecting the sample of input genotypes decomposed into pairs of haplotypes), and the arcs represent potential blocks that are being tested using one of the block identification criteria.

The shaded areas are the parts of the haplotype matrix that are traced twice. The time needed to go through the shaded areas is then bounded by $2*2nm$.

During the test for each potential block, the distinct patterns contained in it are determined. To ensure maximal time saving, this is performed within function *dist* which uses identifiers for every pattern in the original blocks instead of the patterns themselves.

There are 2 blocks proposed for merging and, therefore, there are 2 lists of identifiers – one for each of these blocks.

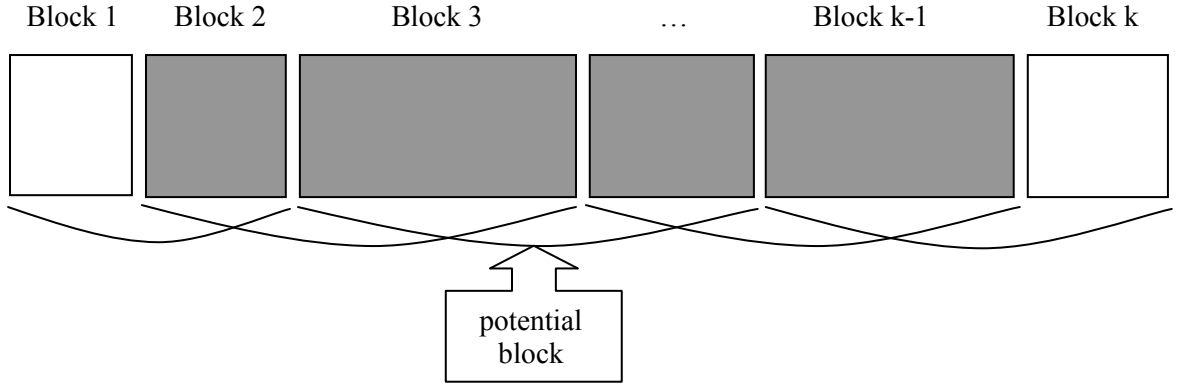


Figure 4.1 Block merging process (as a part of a block-extension algorithm) at each level of a corresponding binary tree.

The distinct patterns for the potential block are created by finding a new list of distinct 2-element sequences (first element from a sequence comes from the first list, second element – from the second list). This process consists of comparing every next pair of ordered identifiers to the list of those already discovered at most **(i-1)** distinct 2-element sequences. It takes time proportional to $\sum_{i=1}^{n_b} 2(i-1) = O(n^2)$ for each potential block. The calculation of either of the two block identification criteria takes time proportional to **(n+l_b)**, where **l_b** is the length of a potential block. Summing over all of the potential blocks (if **k** is the number of current blocks; there are **k-1** of the potential blocks being tested):

$$\sum_{b=1}^{k-1} (O(n^2) + O(n + l_b)) = \left(\sum_{b=1}^{k-1} (O(n^2) + O(n)) \right) + O\left(\sum_{b=1}^{k-1} l_b \right) = O(n^2 m) + O(nm) + O(m). \quad (4.1)$$

Since the sum of the lengths of all tested blocks is bounded by **2m** (by the argument used above), and **k** is bounded by **m**, the above expression is equal to **O(n²m)+O(nm)+O(m) = O(n²m)**. Each of the potential blocks is traced some constant number of times to fill in the newly found distinct patterns and their profiles (frequency and line index) which adds to the overall time **const*2nm**. Thus, the time complexity of each level is **O(nm)+O(n²m)=O(n²m)**. Since, as discussed earlier, the maximal number of

levels is equal to $(m-1)$, the overall time complexity of the block-extension algorithm is $O(n^2m^2)$.

4.3 Time complexity of assessment of the fitness of all haplotype patterns within every block

During the calculation of the fitness for each pattern in a particular block, that pattern (of length equal to the length of a block l_b) is compared element by element to each of the n input genotypes (within the same boundaries). This process, together with filling in the applicability matrix along the way, takes time proportional to l_bn . The number of patterns within each block is bounded by $2n$; therefore, the time needed to perform the fitness calculation over all blocks is proportional to

$$\sum_{b=1}^B l_b n 2n = 2n^2 \sum_{b=1}^B l_b = 2n^2 m \quad (4.2)$$

Thus, the time complexity of this step is equal to $O(n^2m)$.

4.4 Time complexity of the selection of the fittest subset within every block

Consider the random selection of the fittest subset of patterns within each block. Every time the random selection of one pattern out of the set of all available patterns is performed, the list of all patterns (defined by the max number $2n$) is traced once. By taking into consideration filling in respective fractional fitness values for those patterns, the time complexity of this process then becomes $O(n)$. Each selected pattern is then checked for any unmarked genotypes that can potentially be resolved using this pattern. Since there are n genotypes to be searched through, the time complexity of each random selection round is $O(n)+O(n)=O(n)$. The random selection is performed until all genotypes are marked. In the worst case scenario, this is done $2n$ times (all patterns have to be selected). Thus, the time complexity of the selection process within each block is then $2nO(n)=O(n^2)$. The number of blocks is proportional to the length of the SNP sequence, i.e., $O(m)$. Therefore, the overall time spent to perform the selection operation in all blocks is $O(n^2)O(m)=O(n^2m)$.

4.5 Time complexity of the construction of the next generation of haplotypes based on the selected haplotype patterns

The reconstruction of the new generation is performed separately in each block. Within each block for every genotype (total number is n), it is determined which pattern, potentially resolving this genotype, can be used as a base. To do this, the list (of size n_b) of all patterns is searched in those that are selected (during the previous step) and applicable. If there are several of them, random selection proportional to fitness is then performed. One random selection round takes $O(n)$ time as was described in the previous section. After the base resolving pattern is established, the model for the second resolving pattern is created in time proportional to the length of the current block (l_b). The model is then compared element-by-element to each of the remaining applicable to this genotype patterns (whose number is $O(n_b)$); this takes $O(l_b n_b)$ time. Again, if there are several such patterns found, random selection of one of them proportional to the fitness values is performed (in $O(n)$ time). Thus, for every genotype it takes $O(n) + O(l_b) + O(l_b n_b) + O(n) = O(n) + O(l_b n_b)$ time to perform the reconstruction. Since the best bound for n_b can be set at $2n$, the time complexity then becomes $O(n) + O(l_b n) = O(l_b n)$.

Since there are n genotypes within each block, and the reconstruction needs to be done in each block, the overall time complexity of this entire step is $O(n^2 m)$:

$$\sum_{b=1}^B O(n) O(n l_b) = O(n^2) \sum_{b=1}^B l_b = O(n^2) m = O(n^2 m) \quad (4.3)$$

4.6 Time complexity of inter-block transitions

The matching of pairs of haplotypes in two adjacent blocks is done by considering every pair of whole-length haplotypes and running through all blocks for these two haplotypes. For every pair of adjacent blocks, the matching involves the calculation of the four probabilities $P(h_{ia}, h_{jb}) = P_{ij}$ as scalar products of the two respective applicability vectors, divided by the number of genotypes, i.e., $P_{ij} = (app_{ia}, app_{jb})/n$. The time needed to compute every such scalar product is proportional to the length of an applicability vector, which is always equal to n . Since the number of times the matching has to be performed for every two whole-length haplotypes is equal to the number of blocks less 1, and the number of blocks is proportional to the length of the SNP sequence (m), the inter-

block transitions for two haplotypes over the entire length of the SNP sequence takes $O(n)O(m)=O(mn)$.

Time complexity of the inter-block transitions for the entire set of $2n$ whole-length haplotypes is then $O(n)O(mn)=O(n^2m)$.

4.7 Time complexity of the adjustment of block structure

Time needed to adjust the block structure is spent, first of all, on checking whether all of the current blocks still satisfy the block identification criterion (with slightly higher threshold), and if they don't, some time spent on destroying them. Second, the block-extension algorithm is applied on this updated block structure.

The first process, involving calculation of the block identification criteria for each of the blocks and re-initialization of some of the invalid blocks, is essentially equivalent to going over the entire haplotype matrix (of size $2nm$) and two other block profile matrices of the same size. Thus, the time complexity of the first part of the adjustment of the block structure is $O(mn)$.

The second part is simply running the block-extension algorithm twice (first time using the NED measure and second time using the coverage of the common 4-5 haplotype patterns as the block identification criteria). As described earlier, time complexity of the block extension algorithm is $O(n^2m^2)$. Thus, the time necessary to complete the adjustment of the block structure is $O(mn)+O(n^2m^2)=O(n^2m^2)$.

4.8 Time complexity of the evaluation of the current solution

On this step, two quantities are computed based on the current haplotype decomposition and block structure: number of distinct whole-length haplotypes, ds , and the total number of common patterns across all blocks, n_{compat} .

The number of distinct whole-length haplotypes, ds , is calculated using time-economical function *dist* used in the block-extension algorithm. Analogous to working with 2-element sequences, in the case of whole-length haplotypes, it uses $|B|$ -element sequences (where $|B|$ is the number of blocks of order $O(m)$). Each element in such a sequence is an identifier (or index) of some pattern. Similar to the calculation of the 2-

element distinct sequences, the time needed to complete function **dist** in the case of $|B|$ -element sequences is proportional to:

$$\sum_{i=1}^{2n} |B| (i-1) = O(n^2) |B| = O(n^2 m). \quad (4.4)$$

Total number of common patterns across all blocks, *ncompat*, is computed by performing the ordering of patterns within each block. There are at most $2n$ patterns within each block, which are ordered in $O(n^2)$ time. The summation over all blocks (the number of which is proportional to m), yields $O(n^2 m)$.

Thus, the overall time complexity of the evaluation of the current solution is $O(n^2 m)$.

4.9 Time complexity of the operation of mutation

The operation of mutation consists of going through the haplotype matrix and switching pairs of elements at some heterogeneous positions. The fact that the mutation is performed only for those pairs of patterns (within each block) that have frequency less than 10% does not add to the time complexity since it only involves a constant time computation of fractional frequency for each of the patterns. Since the size of the haplotype matrix is $2nm$, the time complexity of the mutation operation is $O(mn)$.

The inevitable consequence of the mutation is the changed variety of patterns within each block as well as the changed allele frequencies. This fact should be reflected in the block structure profile matrices (**B**, listing haplotype patterns within each block, and **Fb**, listing frequencies of these patterns and indices of the patterns used in each row of the haplotype matrix) and in the vector containing allele frequencies at each position of the SNP sequence.

Updating the block structure profile involves calculation of all distinct patterns within each block together with their frequencies. This is done by consecutively taking each pattern from row i in the haplotype matrix and comparing it element by element to already listed distinct patterns whose number is at most $i-1$. Since there are $2n$ haplotypes within each block in the haplotype matrix, and the number of blocks is $|B|$, the time needed to complete this process is proportional to:

$$\sum_{b=1}^{|B|} \sum_{i=1}^{2n} l_b (i-1) = \sum_{b=1}^{|B|} l_b \sum_{i=1}^{2n} (i-1) = mO(n^2) = O(n^2 m) \quad (4.5)$$

Calculation of the updated allele frequencies at each SNP position takes the time needed to simply go over the haplotype matrix once and fill the 2 vectors of length m each. Since the size of the haplotype matrix is $2nm$, the time complexity of this process is $O(mn)$.

Thus, the time complexity of the operation of mutation together with updating of the block structure profile and allele frequencies is $O(n^2 m)$.

Summing over all steps of the algorithm, the overall time complexity of each iteration becomes $O(mn) + O(n^2 m^2) + O(n^2 m) + O(n^2 m) + O(n^2 m) + O(n^2 m) + O(n^2 m^2) + O(n^2 m) + O(n^2 m) = O(n^2 m^2)$.

Since the number of iterations is in linear dependency with the size of the input (mn) , the **total time complexity of the proposed genetic algorithm** is $O(mn)O(n^2 m^2) = O(n^3 m^3)$.

Chapter 5

Results

5.1 Methods for the results evaluation

The following are different measures widely used for examining an algorithms' performance in the area of haplotype inference and block segmentation. Among these are measures of the quality of a solution as well as methods for comparing alternative solutions. All of them will be used to assess the results of the current study.

5.1.1 Error rate I

Error rate I is the simplest measure of the quality of the obtained haplotype resolution. It is mentioned by Stephens *et al.* [57] and used by most of the other researchers. It is represented by the proportion of individuals (genotypes) whose haplotypes were incorrectly inferred. This measure doesn't take into account the closeness of the solutions which could differ in only one or in all ambiguous sites. Thus, this is a very rough error rate.

5.1.2 Error rate II

The original paper by Kimmel and Shamir [83] suggests using the error rate II measure for the evaluation of performance within each block (in which case it is called *block error rate*), but it can also be used for full-length sequences. Denote the two true haplotype resolutions for some genotype \mathbf{g}_i by t_1^i and t_2^i . Also, denote two inferred haplotypes by h_1^i and h_2^i . Then the number of errors in genotype \mathbf{g} can be defined as:

$$e_i = \frac{1}{2} \min \left\{ \left[d(t_1^i, h_1^i) + d(t_2^i, h_2^i) \right], \left[d(t_1^i, h_2^i) + d(t_2^i, h_1^i) \right] \right\}, \quad (5.1)$$

where \mathbf{d} is the *Hamming distance*. The Hamming distance is the number of positions in two strings of equal length for which the corresponding elements are different. Given that the number of heterogeneous sites in genotype \mathbf{g}_i is r_i , the error rate for the entire group of genotypes within specified boundaries is

$$err_{II} = \frac{\sum_{i=1}^n e_i}{\sum_{i=1}^n r_i} \quad (5.2)$$

The R code for the function calculating the hamming distance between the true and computed haplotypes and the function for computing the Error rate I for the entire data set is given in the Appendix A.

5.1.3 Switch rate

The switch rate is possibly a more adequate measure of the haplotype resolution when applied to full-length haplotypes than the error rate II. Its use for matching blocks was suggested by Kimmel and Shamir in [83]. In addition to measuring the quality of a single solution versus the true solution, the switch test can be used to compare two alternative solutions. Let the number of switches s_i between two solutions $t_i = (t_1^i, t_2^i)$ and $h_i = (h_1^i, h_2^i)$ for the same genotype g_i be the minimum number of switches (from 0 to 1 or otherwise) at heterogeneous sites necessary to obtain one solution from the other. Similar to the error rate II (with the number of heterogeneous sites in the genotype being r_i), the *switch rate* for the entire collection of genotypes is defined as

$$SR = \frac{\sum_{i=1}^n s_i}{\sum_{i=1}^n r_i} \quad (5.3)$$

It can be argued that the switch rate is more adequate than the block error rate (error rate II) since the several “errors” can be eliminated by simply considering switch rather than the hamming distance. In an example given by Kimmel and Shamir [83] in Table 5.1, there are 5 errors and only 2 switches:

Table 5.1 Example 1 for comparison of the error rate II and the switch rate.

<i>Computed haplotypes</i>	<i>True haplotypes</i>
1 1 1 1 1 0 0 0 0 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 1 1 1 1 1 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0

In this case, the switch rate seems to provide a more appropriate error rate. On the other hand, consider an example given in Table 5.2. There are 6 switches and only 3 errors. Thus, error rate II would give a better rate.

Table 5.2 Example 2 for comparison of the error rate II and the switch rate.

<i>Computed haplotypes</i>	<i>True haplotypes</i>
1 1 1 1 1 0 1 0 1 0 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 1 0 1 0 1 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0

Due to the reasons described above, the switch rate is mostly used for calculation of the whole-length error rates (when the error increases due to the improper matching of blocks) and the error rate II seems more appropriate for the estimation of the accuracy within blocks.

The function to calculate the switch rate as implemented in the R code is given in Appendix A.

5.2 Results from applying the algorithm to real and simulated data

The output from running the proposed genetic algorithm include: the haplotype matrix for the whole-length haplotypes, the block structure described by the block boundaries and the lists of haplotype patterns within each block and the list of respective frequencies of these patterns. The algorithm also supplies the vector of scores for each SNP as a block boundary, where score is the proportion of times the algorithm selects this position as a block boundary as a part of its current best solutions.

The algorithm was applied to several publicly available sets of data and the results were compared to those from previous studies based on the same data.

5.2.1 Drysdale data

First, we used the data set originally reported by Drysdale *et al.* [97], where 13 variable sites spanning 1.6 kb of the human β_2 AR gene were collected from 121 subjects. Out of these individual genotypes only 18 distinct genotypes were identified and studied. One site did not exhibit ambiguity in the sample and was excluded from consideration. This resulted in the data consisting of 18 genotypes recorded in 12 SNP sites. The original paper [97] gave 12 haplotypes but found that only 10 distinct haplotypes exist in a studied asthmatic cohort. These 10 true haplotypes were also found by the parsimony-based computational algorithm by Wang, Xu [97].

Our algorithm consistently finds a true solution in less than 150 iterations (approximately 2 sec. using 2.3 Ghz processor). In addition, the algorithm sometimes finds another equally valid solution (from the point of view of maximal parsimony). Both solutions have the same number of common haplotypes (providing at least 80% coverage) and the same total number of haplotypes. Therefore, the algorithm does not distinguish between them. Both solutions as well as the original data are given in Table 5.3. The fact that there are two solutions to the problem, both resulting in a minimum number of haplotypes used to decompose the original data, is not uncommon for data of small size. To produce a more reliable result, one has to collect a larger number of genotypes or introduce additional constraints to the model.

Table 5.3 Drysdale data and the solutions found by the genetic algorithm.

	<i>Original data</i>	<i>Resolved patterns</i>	<i>Distinct haplotypes</i>	<i>Haplotype</i>	<i>Frequency</i>
Solution 1 (true)	2 0 2 2 2 2 2 2 0 0 0	(h1, h2)	1 0 0 1 1 1 1 0 1 0 0 0	h1	9
	1 0 0 1 1 1 1 0 1 0 0 0	(h1, h1)	0 0 1 0 0 0 0 1 0 0 0 0	h2	10
	2 0 0 2 2 2 2 0 2 2 0 2	(h1, h3)	0 0 0 0 0 0 0 0 0 1 0 1	h3	4
	0 0 1 0 0 0 0 1 0 0 0 0	(h2, h2)	0 0 1 0 0 0 0 0 0 0 0 0	h4	2
	0 0 2 0 0 0 0 2 0 2 0 2	(h2, h3)	*0 0 0 0 0 0 0 0 0 1 0 0*	h5	1
	2 0 2 2 2 2 2 0 2 0 0 0	(h4, h1)	1 0 0 0 0 0 0 1 0 0 0 0	h6	2
	0 0 2 0 0 0 0 2 0 2 0 0	(h2, h5)	0 0 0 0 0 0 0 0 0 0 0 0	h7	1
	2 0 2 0 0 0 0 1 0 0 0 0	(h2, h6)	0 1 1 0 0 0 0 1 0 0 0 0	h8	2
	2 0 0 0 0 0 0 2 0 2 0 2	(h3, h6)	0 0 0 0 0 0 0 0 0 1 1 1	h9	3
	2 0 0 2 2 2 2 0 2 0 0 0	(h1, h7)	0 0 1 0 0 0 0 1 0 1 0 1	h10	2
	2 2 2 2 2 2 2 2 0 0 0 0	(h1, h8)			
	2 0 0 2 2 2 2 0 2 2 2 2	(h1, h9)			
	2 0 2 2 2 2 2 2 2 2 0 2	(h1, h10)			
	0 2 1 0 0 0 0 1 0 0 0 0	(h2, h8)			
	0 0 1 0 0 0 0 2 0 0 0 0	(h2, h4)			
	0 0 2 0 0 0 0 2 0 2 2 2	(h2, h9)			
	0 0 1 0 0 0 0 1 0 2 0 2	(h2, h10)			
	0 0 0 0 0 0 0 0 0 1 2 1	(h3, h9)			
Solution 2	2 0 2 2 2 2 2 2 0 0 0	(h1, h2)	1 0 0 1 1 1 1 0 1 0 0 0	h1	9
	1 0 0 1 1 1 1 0 1 0 0 0	(h1, h1)	0 0 1 0 0 0 0 1 0 0 0 0	h2	9
	2 0 0 2 2 2 2 0 2 2 0 2	(h1, h3)	0 0 0 0 0 0 0 0 0 1 0 1	h3	4
	0 0 1 0 0 0 0 1 0 0 0 0	(h2, h2)	0 0 1 0 0 0 0 0 0 0 0 0	h4	3
	0 0 2 0 0 0 0 2 0 2 0 2	(h2, h3)	*0 0 0 0 0 0 0 0 1 0 1 0*	h5	1
	2 0 2 2 2 2 2 0 2 0 0 0	(h4, h1)	1 0 0 0 0 0 0 1 0 0 0 0	h6	2
	0 0 2 0 0 0 0 2 0 2 0 0	(h4, h5)	0 0 0 0 0 0 0 0 0 0 0 0	h7	1
	2 0 2 0 0 0 0 1 0 0 0 0	(h2, h6)	0 1 1 0 0 0 0 1 0 0 0 0	h8	2
	2 0 0 0 0 0 0 2 0 2 0 2	(h3, h6)	0 0 0 0 0 0 0 0 0 1 1 1	h9	3
	2 0 0 2 2 2 2 0 2 0 0 0	(h1, h7)	0 0 1 0 0 0 0 1 0 1 0 1	h10	2
	2 2 2 2 2 2 2 2 0 0 0 0	(h1, h8)			
	2 0 0 2 2 2 2 0 2 2 2 2	(h1, h9)			
	2 0 2 2 2 2 2 2 2 2 0 2	(h1, h10)			
	0 2 1 0 0 0 0 1 0 0 0 0	(h2, h8)			
	0 0 1 0 0 0 0 2 0 0 0 0	(h2, h4)			
	0 0 2 0 0 0 0 2 0 2 2 2	(h2, h9)			
	0 0 1 0 0 0 0 1 0 2 0 2	(h2, h10)			
	0 0 0 0 0 0 0 0 0 1 2 1	(h3, h9)			

5.2.2 ACE data

Data originally reported by Rieder *et al.* in [98] represents the gene DCP1 sequence collected from 11 subjects (6 from the European-American and 5 from the African-American populations). The gene encodes the angiotensin converting enzyme (hence the name ACE of the data), participating in the regulation of the fluid-electrolyte balance and systemic blood pressure. The ACE data originally consisted of the 78 variant sites (SNPs), out of which only 52 were non-unique polymorphic sites, which were then used in the subsequent analysis. 13 haplotypes resolving 11 genotype sequences were identified and then verified using allele-specific PCR. Several haplotyping studies have applied computational algorithms to analyze these data.

Table 5.4 shows results from running the algorithm on $(3 \times 250) = 750$ iterations (less than 4 sec. on 2.3 Ghz processor) on the ACE data using two sets of selection criteria: a single criterion (the minimum number of distinct whole-length haplotypes (*ds*)) and a double criterion (the minimum number of distinct whole-length haplotypes (*ds*) and the minimum total number of common patterns across all blocks (*ncompat*)). The table indicates that better results are produced using the single criterion (min *ds*) as opposed to the double criterion. These results are compatible with those shown by other studies [65], where the average error rate for different algorithms was 0.27 with very few algorithms [65] producing a 0.18 error rate. The advantage of the genetic algorithm is that it can achieve the same (or better) accuracy compared to most algorithms and does it in a very competitive time interval, while in addition providing a block structure .

Table 5.4 Result of running the algorithm on the ACE data using different optimization criteria.

	<i>Min ds</i>	<i>Min ds & Min ncompat</i>
Number of correctly resolved genotypes	8 or 9 out of 11	7 or 8 out of 11
Error rate (proportion of correctly resolved genotypes)	0.18 – 0.27	0.36 – 0.27
Total number of haplotypes found	13 – 14	14
Number of true haplotypes found	9 out of 13	8 or 9 out of 13

Due to the lack of information, further improvement does not seem possible. In addition to showing a low error rate, the genetic algorithm also consistently finds (as the most frequent outcome) the block structure: block 1 spanning positions 1–14, block 2

spanning positions 15–52. Fig. 5.1 illustrates the outcomes of the scores for block boundaries from the 4 successive runs of the algorithm. The clear pattern can be observed from the graphs: the highest block score usually occurs at the 15th position (the first position always has a high score just by way of calculation). This certainly supports the conclusion about the block structure for these data.

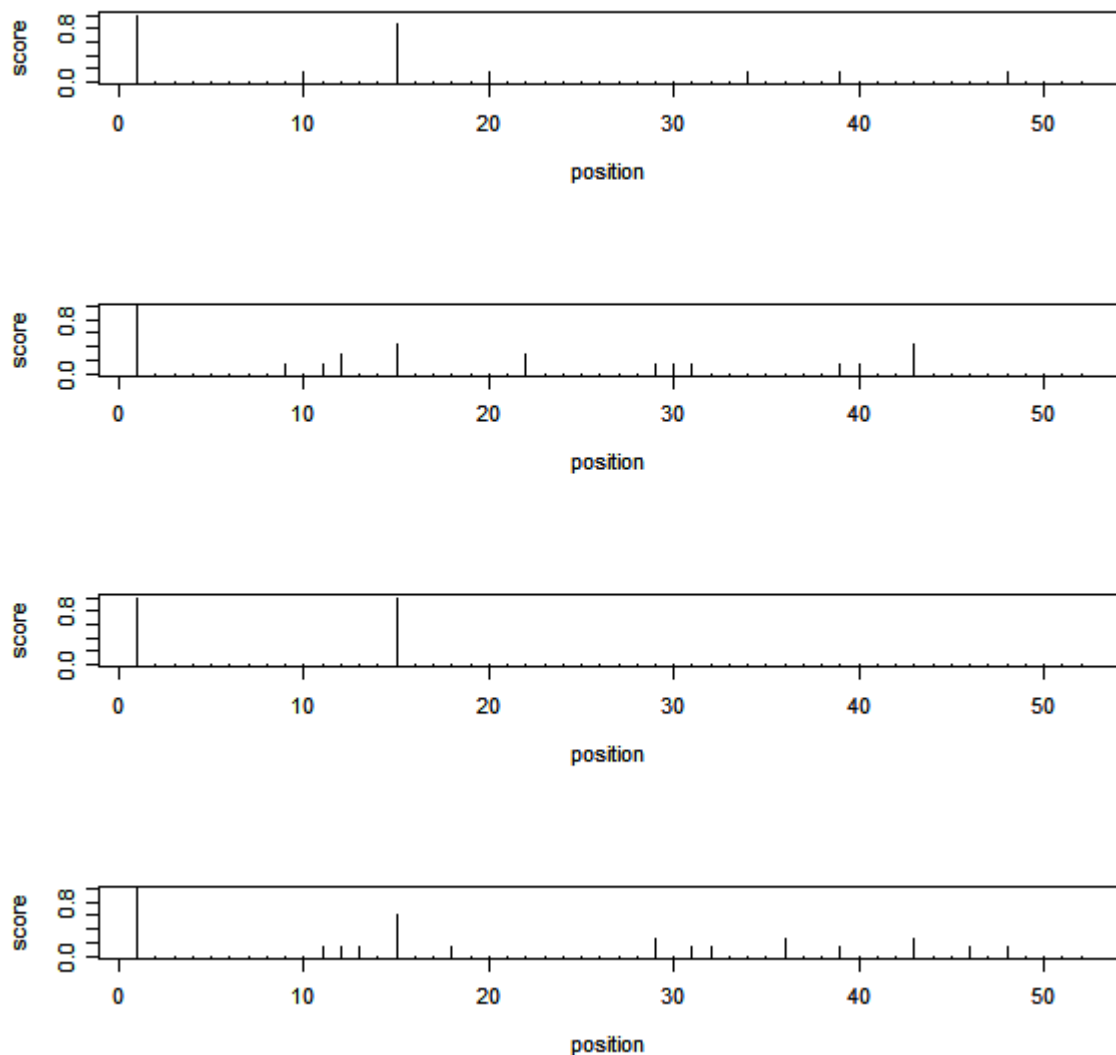


Figure 5.1 Block boundaries scores for the 4 successive runs of the algorithm on the ACE data.

5.2.3 Daly data

The most popular data set widely used in haplotyping and block partitioning algorithms is the one introduced in Daly *et al.*, 2001 [6]. These data were collected over a 500 kilobase region of chromosome 5p31 for 103 SNPs obtained from 129 mother, father and child trios in a European-derived population.

The Mendelian hereditary laws can be applied to infer the exact haplotype decomposition for the majority of sites for each child in a trio. In the original paper the children's data were processed that way and then analyzed for block partitioning. Then using a Hidden Markov Model, the authors split 103 SNPs into 11 blocks separated by intervals where historical recombination events seem to have occurred. According to the results of Daly *et al.* each block contained 5 to 31 consecutive SNPs ranging from 3 to 92 kilobases. The 129 child genotypes, together with corresponding pairs of haplotypes and block structure, comprised the data set of unrelated individuals, which can be used for the haplotype inference problems as well as block partitioning.

Raw Daly data representing two family trios (PED054 and PED058) as given in the original file is

PED054	430	0	0	1	0	1	3	3	1	4	1	4	2	2	1	3	1	2	4	...
PED054	412	430	431	2	2	1	3	1	3	4	1	4	2	2	1	3	1	4	2	...
PED054	431	0	0	2	0	3	3	3	3	1	1	2	2	1	1	1	1	2	2	...
PED058	438	0	0	1	0	3	3	3	3	1	1	2	2	1	1	1	1	2	2	...
PED058	470	438	444	2	2	3	3	3	3	1	1	2	2	1	1	1	1	2	2	...
PED058	444	0	0	2	0	3	3	3	3	1	1	2	2	1	1	1	1	2	2	...
...

Columns 2-4 represent ID numbers: own ID, Father ID and Mother ID. The children in the two families therefore have ID's 412 and 470 respectively. The genotype string of length 108 begins at the 7th column where each position is represented by the two numerically encoded nucleotide bases (without any particular ordering).

Preprocessing of the Daly data using the Mendelian hereditary laws was performed as follows: according to the Mendelian law a child inherits one chromosome from each parent. In the raw Daly data genotypes were encoded by digits 1, 2, 3, 4 corresponding to the four nucleotide bases A, C, G, T, respectively. For ease of treatment, the pairs of numerically encoded nucleotide bases corresponding to each SNP site were ordered in an increasing manner. There were several cases that may have occurred in the data during the preprocessing step. All of the cases for Child's heterogeneous positions

together with their unique resolution (if such is possible) are given in Table 5.5. The numerical encodings are just particular examples, i.e., encoding “13” may also be replaced by “12,” “14,” “24” or “34” and simply stands for the heterogeneous site. Case 1 describes the situation when one of the parents is heterozygous and another is homozygous. The final resolution is always possible in this case and is given in the order that respects the parents ordering, i.e., the first position always corresponds to the Father’s and the second to the Mother’s chromosome. The same ordering is valid for the rest of the cases. Case 2 deals with the situation when both parents are homozygous which is always possible to resolve uniquely. Case 3 occurs when one parent is homozygous and another has missing data (given by “00”). This is also a completely resolvable situation. Cases 4, 5 and 6 occur when each parent has either missing information or is heterozygous. Neither of these cases is resolvable. Code for the Daly data processing is given in Appendix B. Unresolvable cases are marked “*” in the true Child resolution.

Preprocessing of the Daly data was completed by changing paired nucleotide encoding into “0” or “1” (homogeneous positions), “2” (heterogeneous positions) and “9” (missing data) codes. Despite the fact that the family trios can be used to infer the haplotype information on the children, it was not possible to do this for all SNP sites: even after this process was performed, i.e. about 16% of the entire data were either missing or could not be uniquely resolved (ambiguous).

Table 5.5 Resolution of the children genotypes of the Daly data at heterogeneous positions into two haplotypes using parents’ genotype information.

	Case 1	Case 2	Case 3	Case 4 unresolvable	Case 5 unresolvable	Case 6 unresolvable
Father	13 or 13 or 11 or 33	44 or 22	22 or 00	00 or 12	00	13
Mother	11 33 13 13	22 44	00 22	12 00	00	13
Child	13 13 13 13	24 24	24 24	12 12	13	13
Child true	31 13 13 31	42 24	24 42	*	*	*

The proposed genetic algorithm was then applied to the preprocessed Daly data with the double optimization criterion (*min ds & min ncompat*) which has shown better performance for larger data sets than the single *min ds* criterion.

Table 5.6 Haplotype patterns within Daly block partition: original and predicted by the algorithm.

<i>Blocks</i>	<i>Predicted patterns</i>	<i>Original patterns (Daly et al.)</i>
Block 1 1 – 8	"GGACAACC" "AATTCGTG"	"GGACAACC" "AATTCGTG"
Block 2 10 – 14	"TTACG" "CCCAA"	"TTACG" "CCCAA"
Block 3 16 – 24	"CGGAGACGA" "CGCAGACGA" "GACTGGTCG" "CGGATACGA"	"CGGAGACGA" "CGCAGACGA" "GACTTGTCG"
Block 4 25 – 35	"CGCGCCCGGAT" "CTGCTATAACC" "CTGCCCCGGCT" -- "TTGCCCCAACC" --	"CGCGCCCGGAT" "CTGCTATAACC" "CTGCCCCAACC" "TTGCCCCGGCT"
Block 5 36 – 40	"CCAGC" "CCACC" "GCGCT" "CCGCT" --	"CCAGC" "CCACC" "GCGCT" "CAACC"
Block 6 41 – 45	"CCGAT" "CTGAC" "ATACT"	"CCGAT" "CTGAC" "ATACT"
Block 7 46 – 76	"CCCTGCTTACGGTGCAGTGGCACGTATT*CA" -- "TCCCATCCATCATGGTCGAATGCGTACATTA" "CCCCGCTTACGGTGCAGTGGCACGTATATCA"	"CCCTGCTTACGGTGCAGTGGCACGTATT*CA" "CATCACTCCCCAGACTGTGATGTTAGTATCT " "TCCCATCCATCATGGTCGAATGCGTACATTA" "CCCCGCTTACGGTGCAGTGGCACGTATATCA"
Block 8 78 – 84	"CGTTTAG" "TGTT*GA" "TGATTAG" "CGTCTAG" --	"CGTTTAG" "TGTT*GA" "TGATTAG" "TAATTGG"
Block 9 86 – 91	"ACAACA" "GCGGTG" "ACGGTG" "GTGACG"	"ACAACA" "GCGGTG" "ACGGTG" "GTGACG"
Block 10 92 – 98	"GTTCTGA" "TG*GTAA" "TGTGCGG"	"GTTCTGA" "TG*GTAA" "TGTGCGG"
Block 11 99 – 103	"CGGCG" "TATAG" "TATCA"	"CGGCG" "TATAG" "TATCA"

The "--" symbol indicates that the true pattern is either missing or some other pattern found instead.

When running the algorithm separately on each of the original Daly blocks, it correctly predicts most common patterns (covering at least 90%) as seen from Table 5.6.

The incorrect prediction corresponds to only 5 patterns, and 1 pattern in block 3 is predicted in addition to those determined by Daly for this block.

Table 5.7 Typical outcome from running the algorithm for the Daly data

<i>Block (positions)</i>	<i>Total number of patterns</i>	<i>Common Patterns</i>	<i>Coverage</i>	<i>Error rate</i>
1 through 14	25	"GGACAACCGTTACG" "AATTCGTGGCCCAA" "GGACAACCTTTACG"	77.34% 7.42% 3.90%	0.00222
15 through 20	20	"CCGGAG" "CCGCAG" "TCGCAG" "TGA CTG"	46.48% 14.45% 5.47% 19.92%	0.01785
21 through 28	15	"ACGACGCG" "ACGACTGC" "GTCGCTGC" "GTCGTTGC"	55.46% 17.57% 15.62% 5.08%	0.00278
29 through 37	22	"CCCGGATCC" "TATAACCGC" "CCCGGCTCC" "CCCAACCCC"	54.3% 10.94% 16.01% 6.25%	0.02209
38 through 46	20	"ACCCTGATC" "GCTCTGACT" "ACCATACTC" "ACCCTGACT" "AGCCCGATC"	3.91% 13.28% 10.55% 7.42% 53.52%	0.02424
47 through 71	32	"CCTGCTTACGGTGCAGTGGCACGTA" "CCCATCCATCATGGTCGAATGCGTA" "CCCGCTTACGGTGCAGTGGCACGTA"	57.81% 22.27% 9.38%	0.00246
72 through 80	36	"TTGCACCGT" "CATTACTGT" "CATTAGTGT"	54.30% 10.94% 8.98%	0.03438
81 through 87	24	"TTAGCAC" "TTGACGC" "TTGAGGC" "CTAGCAC"	61.72% 14.84% 5.47% 3.91%	0.04188
88 through 94	24	"AACAGTT" "AACATGT" "GGTGTGT" "GACGTGT" "GGTGTGC"	48.83% 6.64% 15.23% 7.03% 7.42%	0.02564
95 through 103	23	"CTGATATAG" "GCGGCGGCG" "GTAACGGCG" "CTGACGGCG" "GCGGTATCA"	40.23% 7.42% 21.09% 12.89% 6.25%	0.01342
Average block error rate: 0.01523				

In general, block structure obtained as a result of running the algorithm varied from one run to another, but there were consistencies. One of the typical outcomes is shown below in Table 5.7. The accuracy of within block haplotype decomposition was assessed using the block error rate discussed above.

The accuracy of the whole-haplotype decomposition is represented by the switch rate (performed on haplotypes assembled out of the separate block patterns). Table 5.8 shows the average switch rate (based on 10 successive runs) for the *HAPLOGEN* algorithm and the average block error rate for the Daly data compared to the same measures of the other four state-of-the-art algorithms (*fastPHASE*, *GERBIL*, *HAP*, *HaploBlock*). All parameters of these algorithms were taken at the default values. Table 5.8 also provides information on the average running time for different algorithms when they are run on the 2.3 Ghz processor.

Table 5.8 Performance of the proposed genetic algorithm compared to the other four algorithms for phasing and block partitioning.

	<i>fastPHASE</i>	<i>GERBIL</i>	<i>HAP</i>	<i>HaploBlock</i>	<i>HAPLOGEN</i>
Ave.Bl.Err.Rate	-	0.0067	0.0119	0.0178	0.0158
Switch rate	0.019	0.0297	0.0421	0.0323	0.0427
Running time	6 m. 35 s.	0 m. 57 s.	*	over 8 h.	3 m. 40 s.

* In was not possible to obtain the running time for the HAP algorithm.

FastPHASE algorithm does not produce a block structure even though it models linkage disequilibrium patterns. It was included into the comparison since it is the best, most recent algorithm for haplotype resolution. Table 5.8 shows that the proposed algorithm *HAPLOGEN* exhibits accuracy within the range of the rest of the algorithms in the “block-based” group (*GERBIL*, *HAP*, *HaploBlock*). The best accuracy was shown by the *fastPHASE* algorithm, while the *GERBIL* algorithm was the fastest one.

Scores for the block boundaries calculated for the Daly data in 4 successive runs of the algorithm are shown in Fig. 5.2. These vectors consistently show high scores for the same block boundaries independently of the final outcomes for the block structure (shown by black dots at the top of each graph). According to these scores, the block structure for the Daly data is represented by the following strong boundaries (indicating starting positions of the blocks): 1 – 15 – 21 – 29 – 38 – 45 – 82 – 88 – 95.

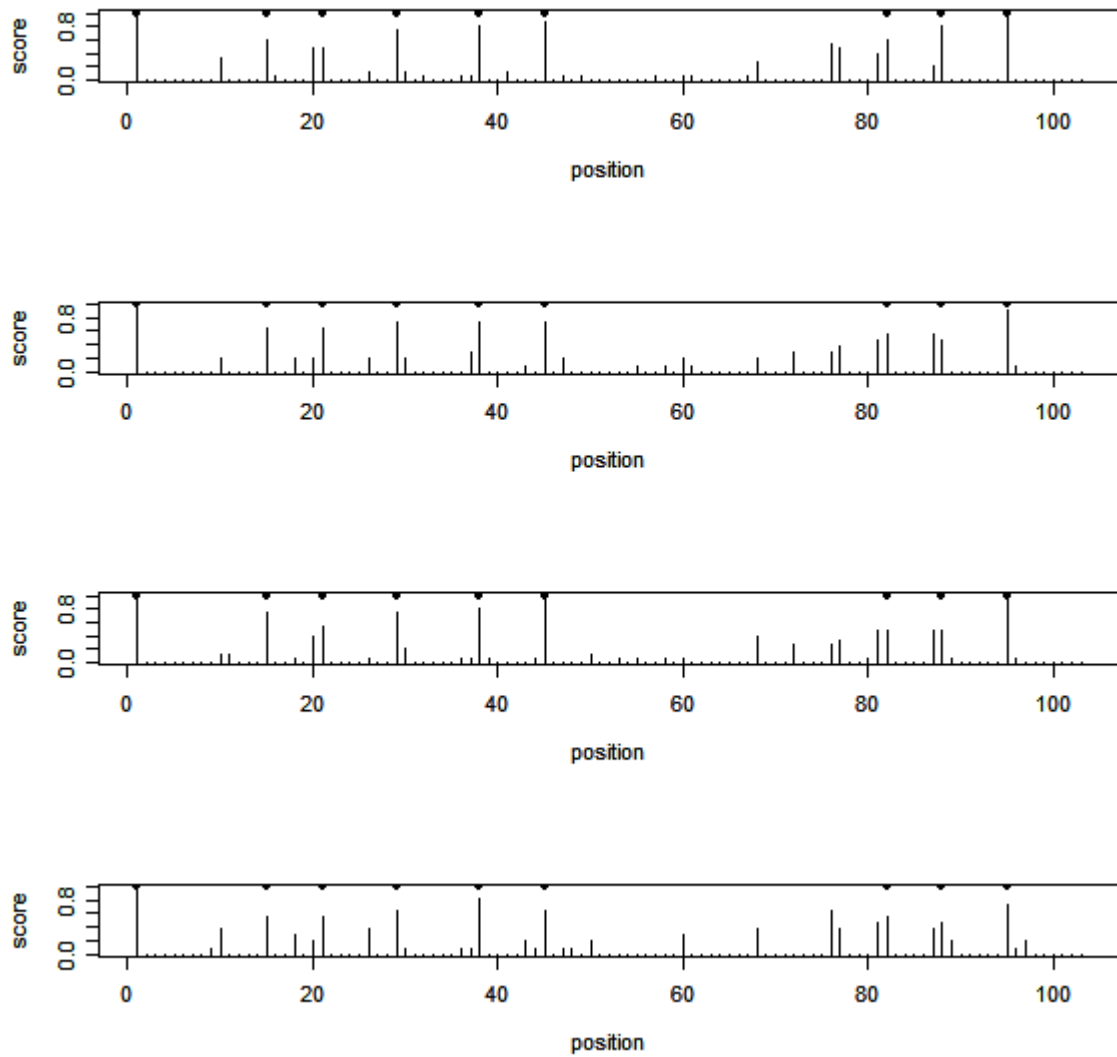


Figure 5.2 Scores for block boundaries from the 4 successive runs of the algorithm (dots at the top of each graph indicate final, best solution for the block structure).

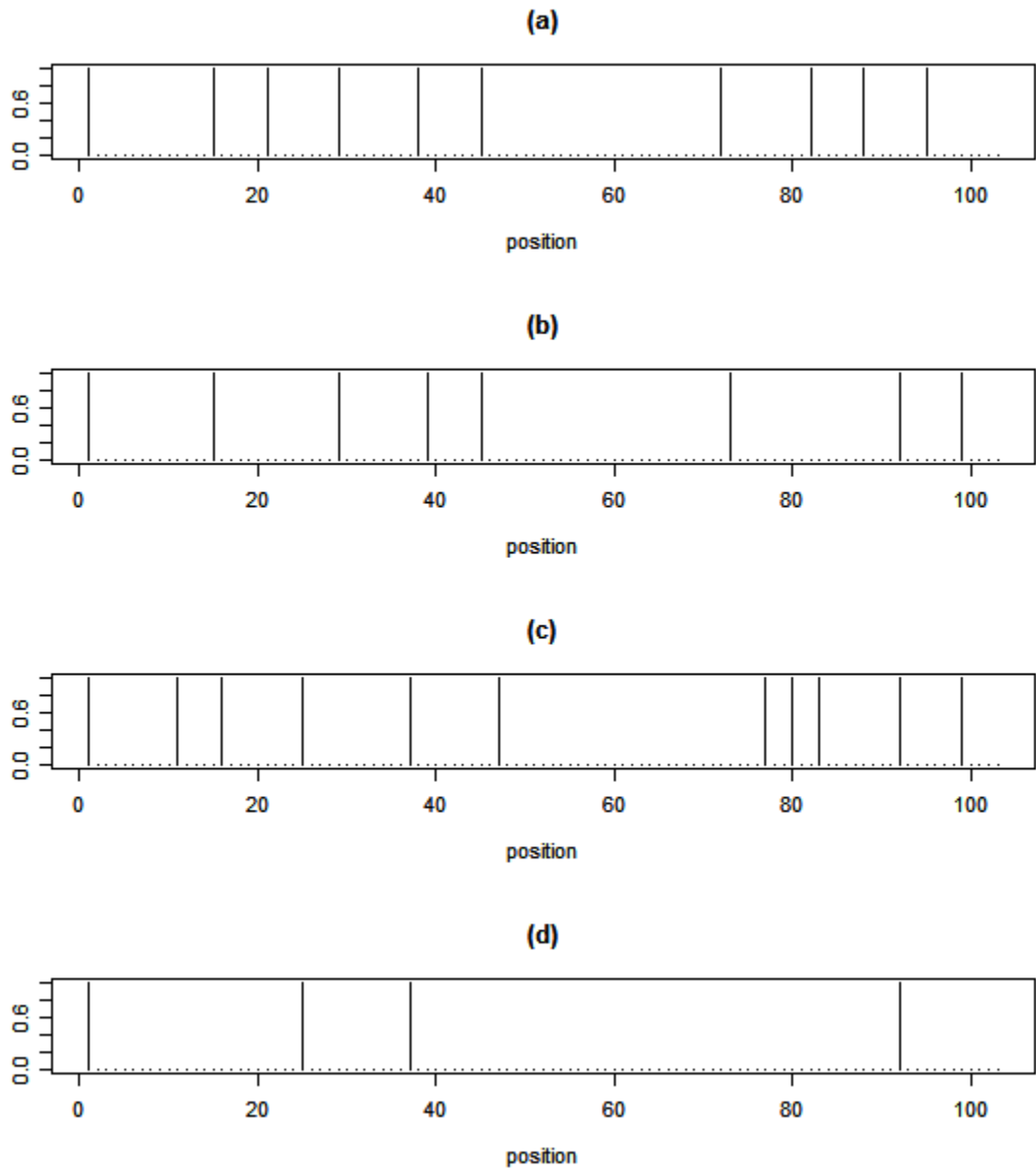


Figure 5.3 Comparison of the block boundaries for the Daly data from different algorithms: (a) *HAPLOGEN*, (b) *GERBIL*, (c) *HAP*, (d) *HaploBlock*, (e) original Daly partition.

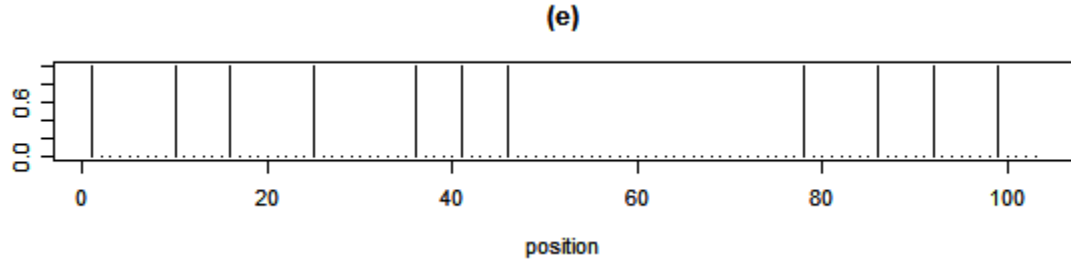


Figure 5.3 (continued) Comparison of the block boundaries for the Daly data from different algorithms: (a) *HAPLOGEN*, (b) *GERBIL*, (c) *HAP*, (d) *HaploBlock*, (e) original Daly partition.

The proposed genetic algorithm shares 3 boundaries (not counting the beginning of the first block) with the *GERBIL* algorithm and no boundaries with any other, but several boundaries in each case are very close to each other (difference in 1-2 position) as can be seen in Fig.5.3 that gives block boundaries for the Daly data from different algorithms. Counting together with the close positions, all algorithms share the number of positions given in Table 5.9.

Table 5.9 Number of shared block boundaries (including close positions) for the Daly data produced by different algorithms.

	<i>GERBIL</i>	<i>HAP</i>	<i>HaploBlock</i>	<i>Daly</i>
<i>HAPLOGEN</i>	5	3	1	4
<i>Gerbil</i>		5	2	4
<i>Eskin</i>			3	8
<i>HaploBlock</i>				3

The proposed genetic algorithm exhibits comparable accuracy (though not exceeding) and block structure similar to those of the existing phasing and block partitioning algorithms. As an advantage, the genetic algorithm predicts missing data as a part of an algorithm (unlike the other algorithms).

Missing data prediction was performed on the simulated data as follows: output haplotype matrix for the Daly data was turned into the genotype matrix and then a certain percentage of the data was randomly selected to be “missing.”

Table 5.10 Prediction errors
for the missing data

<i>Percentage of missing data</i>	<i>Error rate for prediction of missing data positions</i>
5%	15.5%
10%	19.2%
20%	20%
30%	21.5%

reasonable prediction errors.

Missing data error is calculated as the number of sites non-matching to the true data between switches (as defined for the switch rate) divided by the total number of missing data sites. This is a very strict error rate since it will classify as error even positions with 1 of the 2 haplotypes resolved correctly. Error rates for the different proportions of missing data are given in Table 5.10. These results show that even when there is a significant amount of data missing the algorithm gives

5.2.4 Patil data

Another data set used in block partitioning and haplotyping studies was provided by Patil *et al.*, 2001 [2]. This data set includes the genotype information from the entire chromosome 21 for 24,047 SNPs for which 20 haplotypes were identified by a rodent-human somatic cell hybrid technique (although no genotypic data is provided). Several methods were applied to infer the block structure for this data [2, 11] by using different criteria for the block identification. The data was downloaded from the Perlegen Sciences, Inc. web site. Raw data for the first two blocks, as given in the original file, has the following appearance:

```

block_id pattern_id sample_id haplotype_string
B000001      100   CPD0007C28 ttatnttctngtccgcggggncacgctattcngcga...cnnc
B000001      101   CPD0003C04 catcagctagcattattactttgtctccccgaatag...tgat
B000001      100   CPD0002C28 ttancttcnngtnnnngnngncncnctattccgcga...catc
B000001      100   CPD0004C49 nnnnnnnnnnnnnnnncgnnggcacgctattccgcga...catc
...
B000002      200   CPD0007C28 nntcacancnnnnnnnnnnnnctnngnngngncnnn...nnnc
B000002      200   CPD0003C04 aaaagctgtnnnnnnnnnnnnctacgaatcngnatcac...nnnt
B000002      200   CPD0002C28 cgtcacaacnnnnnnnnnnnnctntatggggagccttt...nnnc
B000002      200   CPD0004C49 cgtcacaacnnnnnnnnnnnnntanggggagccttn...nnnn
B000002      200   CPD0007C08 aaaagctgtnnnnnnnnnnnnctacgaatcagaatcac...nnnt
...

```

Similarly to the previous studies, the Patil data was used here to construct an artificial genotype population. This was done as follows: 100 genotypes were generated by randomly pairing 22 haplotypes. Then the data were analyzed by applying the algorithm to segments of different length (100, 519 and 3295 positions). R code for the

Table 5.11 Switch rates for the simulated Patil data

<i>Length of a segment, m</i>	<i>Switch rate</i>
100	0%
519	1.6%
3295	4.8%

processing of the raw Patil data is given in Appendix C. The accuracy of the prediction of heterogeneous sites is given in Table 5.11. The switch rate was selected to be the more appropriate measure of the error rate rather than the average block error.

The algorithm consistently shows high block error rates (for example, 7.3% for the 100 positions segment), which can be explained by the way the data was simulated. For every genotype, the true haplotype matrix may contain missing data in the first haplotype and non-missing data in the second one (while in the most real data usually both haplotypes will have missing data); in the calculated genotype matrix, the resulting genotype will be recorded as entirely missing. When the hamming distance used to calculate the number of errors is calculated, the missing data is ignored for the first haplotype (when compared to the true solution), but is taken into consideration for the second haplotype). Given that the switch rate is 0, the error rate of 7.3% essentially gives the prediction error for the missing data for part of the haplotypes. Thus, for these data, the average block error rate is not an adequate indicator of error and the switch rate should rather be used.

In general, the algorithm is designed in such a way as to allow the unlimited amount of data which assumes that there should be no problem processing long sequences like, for example, the entire 24,047 SNPs of the Patil data. However, since we are using an R package, the algorithm does have restrictions on the amount of input data. Because of the limitations of the R software, the maximum amount of data the proposed genetic algorithm was able to process was around 350000 entries (number of genotypes times the length of the SNP sequence).

Minor modification can be done in order to improve the speed for the long segments of SNPs. Application of the genetic algorithm to such data at once for the same number of iterations as for the shorter sequences (assuming that the size of genotype sample, n , is the same) should cause loss of accuracy; on the other hand, since the time complexity of the algorithm is $O(mn)O(m^2n^2)$, increasing the number of iterations proportional to the data size mn raises running time by $O(m^3)$ instead of $O(m^2)$. The

following modification to the main algorithm is proposed to solve this problem. Given that the number of iterations only depends on the number of genotypes (n), the low error rate for 500-positions data suggested splitting the entire Patil data into the approximately 500-positions length segments and separately running the algorithm on each segment (using the same number of iterations). The final step is the “gluing” of all solutions consecutively together without loss of accuracy of haplotype resolution as well as loss of local block structure information. This was performed by applying the inter-block transition step to every two adjacent 500-positions segments in order to provide the appropriate matching between the last block(s) of the first segment and the first block(s) of the second segment. The only drawback of the glueing process is that the block structure is only valid within the original 500-position segment boundaries.

The proposed modification by means of splitting the data into 500-position segments was applied to the simulated Patil data spanning 3295 positions. Two versions of the algorithm (original and modified), both based on 1000 iterations (a little over 3 hrs. on a 2.3 Mhz processor), were applied to the data. The average switch rate for the original version of the algorithm was 4.7%, and 4.35% for the modified version. While there is improvement in the accuracy of prediction using the modified version of the algorithm, this reduction in the error rate cannot be considered significant.

5.2.5 HapMap data

The efforts of the International HapMap project (launched in October, 2002) in determining genotypic variations and the common haplotype patterns in the human genome made huge amounts of genotypic data freely available to the public. The International HapMap Consortium provided an elaborate description of the data and a statement of its goals in [69, 70]. The principal goal of the International Consortium is to develop “a map of the haplotype patterns across the genome by determining the genotypes of one million or more sequence variants, their frequencies and the degree of association between them, in DNA samples from populations with ancestry from parts of Africa, Asia and Europe” [69].

The HapMap data were collected from four different populations in the regions of Africa, Asia and Europe. A total of 270 individuals have contributed their DNA samples.

The Yoruba people of Ibadan, Nigeria, provided 30 sets of samples (trios) from two parents and a child; this data set is abbreviated by YRI. Individuals were required to have four of four Yoruba grandparents. The Japanese sample of 45 unrelated individuals from the Tokyo area (referred to as JPT) was collected in such a way that the donors were just asked to have ancestors from Japan. Chinese data consist of the DNA samples from 45 unrelated individuals from Beijing (known as the Han Chinese, abbreviated by CHB), each of whom was required to have at least three of four Han Chinese grandparents. In the US, 30 trios have provided samples (abbreviated by CEU), which were collected in 1980 from U.S. residents with northern and western European ancestry by the Centre d'Etude du Polymorphisme Humain (CEPH). In the CEPH sample there was no specific requirement except for residency in Utah. The collected DNA data are being genotyped in ten centers in Japan, the United Kingdom, Canada, China and the United States using five different genotyping technologies.

In addition to the genotypic data on each human chromosome from each population [99], the HapMap project web-site provides much information on the genome, like the LD map, tag SNP data, SNP allele frequencies, genotype frequencies and also phased haplotype data. The phasing was done using the PHASE software, and compiled from the genotype data to date. The program PHASE implements methods for estimating haplotypes from population genotype data described in [57, 100]. It should be noted that the phasing process used the trio information where available (CEU and YRI samples) so that the resulting published haplotypes were obtained with extremely high degree of accuracy [101].

Thus, the data available from the HapMap web site [99] that were appropriate for the analysis performed in this dissertation included the full chromosome genotypes and computationally derived haplotypes for each of the following population samples:

- Yoruba in Ibadan, Nigeria (YRI) – 30 individuals
- Japanese in Tokyo, Japan (JPT) – 45 individuals
- Han Chinese in Beijing, China (CHB) – 45 individuals
- CEPH (Utah residents with ancestry from northern and western Europe) (CEU) – 30 individuals

The data from the four samples to be analyzed were taken from a randomly chosen chromosome (chromosome 2) in release 22. The 500 SNPs having the same location (first 500 positions of the data) along the chromosome were processed by the *HAPLOGEN* algorithm. The results on the accuracy of the *HAPLOGEN* algorithm haplotype output with respect to those provided by HapMap (using *PHASE* software and the information derived from trios) for each data set are represented in Table 5.12. The estimation of the accuracy rate for the comparison to the true haplotypes can only be made based for the CEU and YRI data. This is due to the fact that only these data contained trios information allowing correct haplotype inference for a lot of SNPs. The HapMap Consortium reported extremely low estimated switch rate (error occurs every 8Mb in CEU and every 3.6Mb in YRI as given in [70]) for the CEU and YRI data when *PHASE* software was used in addition to the family-based haplotype inference. Since the amount of data used for the analysis here does not exceed 600 kb, the switch rate can be regarded as negligible. Thus, the true switch rates calculated using the phased genotypes provided by HapMap can be considered approximately equal to 4.1% and 13.7% for the CEU and YRI data, respectively. Similarly, the true average block error rates are 2% and 7.8%. The discrepancy in haplotype resolutions between *HAPLOGEN* and *PHASE* is estimated around 5% as given by the switch or the average block rates for the CHB and JPT data.

Table 5.12. Performance of the HAPLOGEN algorithm on the HapMap phased data on the 500-positions region of chromosome 2 in different population samples

Data	Average block error rate	Switch rate
CEU	0.02	0.041
YRI	0.078	0.137
CHB	0.05	0.046
JPT	0.055	0.054

Table 5.13. Comparison of the switch rates of several algorithms for haplotype inference for the 500-positions region of chromosome 2 of CEU and YRI data

Data	<i>fastPHASE</i>	<i>GERBIL</i>	<i>HAP</i>	<i>HaploBlock</i>	<i>HAPLOGEN</i>
CEU	0.033	0.029	0.058	-	0.041
YRI	0.0315	0.038	0.1102	-	0.137
Running time:					
CEU	7 m. 26 s.	4 m. 35 s.	*	over 2 h.	5 m. 35 s.
YRI	7 m. 28 s.	5 m. 08 s.		over 2 h.	5 m. 25 s.

* In was not possible to obtain the running time for the HAP algorithm.

Four state-of-the-art algorithms (*fastPHASE*, *GERBIL*, *HAP*, *HaploBlock*) were also applied to the same data for two populations, CEU and YRI, to compare their performance to the proposed algorithm *HAPLOGEN*. All parameters of these algorithms were taken at the default values. The results of the switch rates and the running times (using the 2.3Ghz processor) provided by these algorithms are given in Table 5.13. It took more than 2 hrs. for the *HaploBlock* algorithm to complete the haplotype resolution for each of the samples. Since the rest of the algorithms including the proposed algorithm *HAPLOGEN* provided solutions much faster, the accuracy from *HaploBlock* was not reported here. Table 5.13 shows that *HAPLOGEN* provided switch rate for the CEU sample within the range of the switch rates of the other algorithms. On the other hand, the YRI sample produced relatively high switch rates for the two algorithms *HAP* and *HAPLOGEN*, while the best results were shown by *fastPHASE* and *GERBIL*.

The availability of the haplotype block information through the Haploview software (supplied by the HapMap web site [99]) makes it possible to perform the comparison of the block partitions under different criteria as well as the *HAPLOGEN* outcome. The Haploview provides three commonly used block definitions and the associated block structures: confidence intervals (due to Gabriel *et al.* [15]), four gamete rule (due to Wang *et al.* [12]) and solid spine of LD (due to Barrett *et al.* [101]). The haplotype block structures within the first 170-200 kb of the Chromosome 2 (corresponding to 141-147 SNP positions) obtained from the *HAPLOGEN* and from applying all three criteria for each of the data (CEU, YRI and the combined Asian panel JPT+CHB) are shown in Figures 5.4 through 5.6.

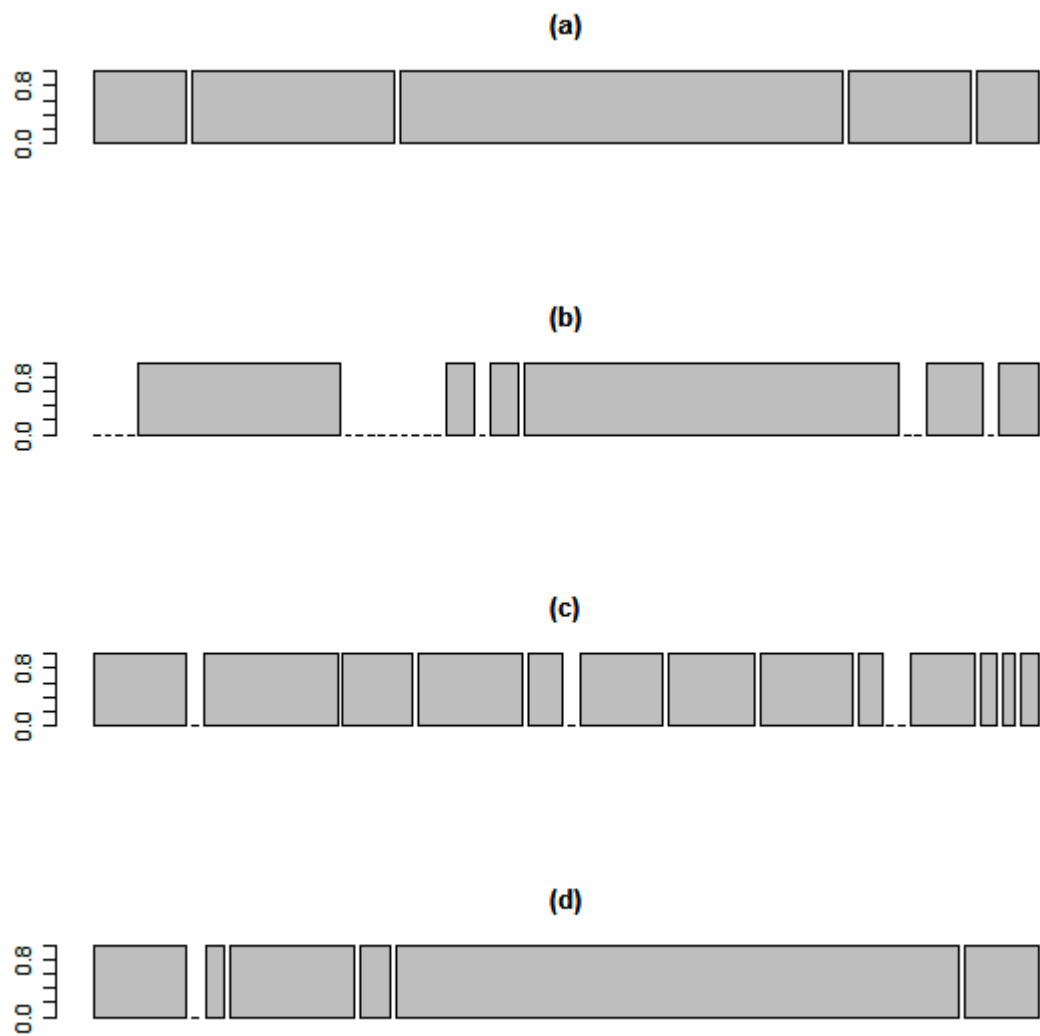


Figure 5.4 Block structure obtained for the CEU data from (a) *HAPLOGEN*, (b) confidence intervals, (c) four gamete rule and (d) solid spine of LD.

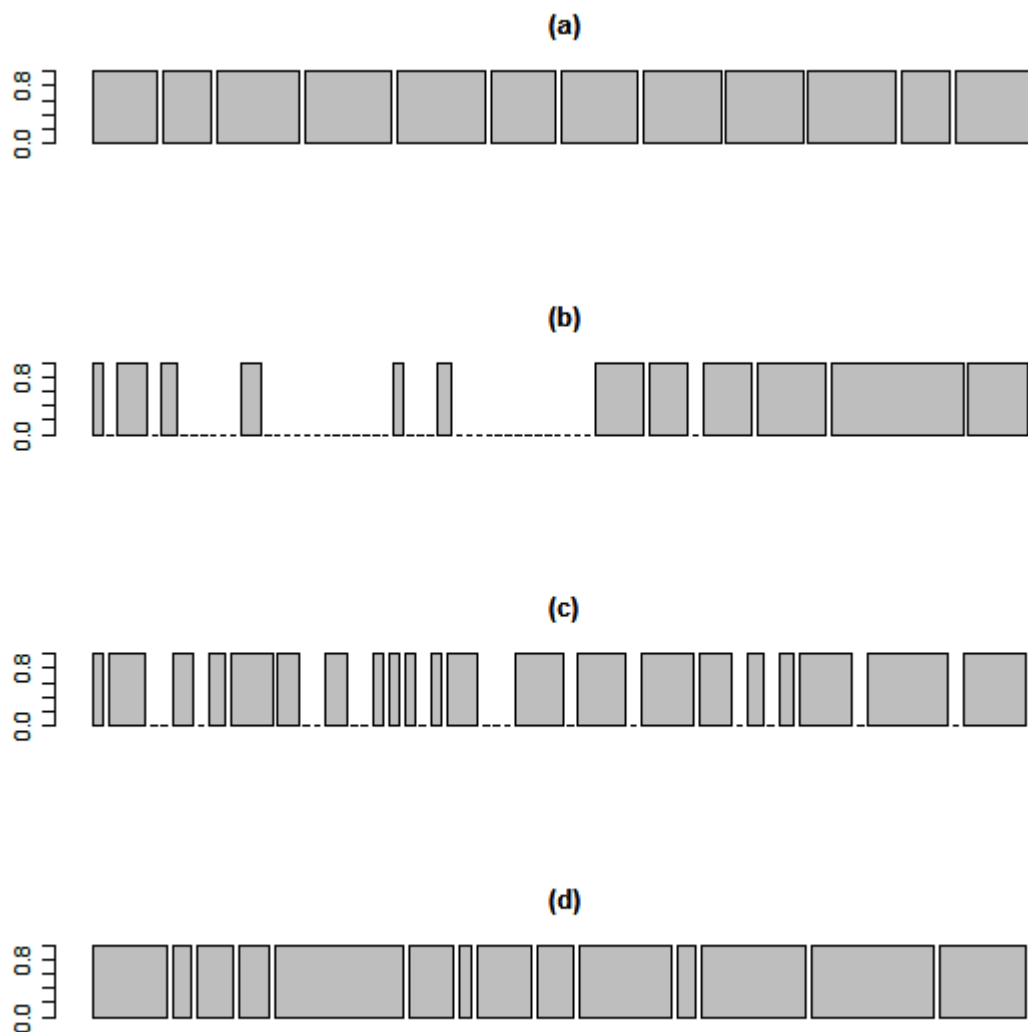


Figure 5.5 Block structure obtained for the YRI data from (a) *HAPLOGEN*, (b) confidence intervals, (c) four gamete rule and (d) solid spine of LD.

The reason that there are no block structures available for each of JPT and CHB is that the Asian data reported in Haploview LD map show up only as the pooled sample (JPT+CHB) and not separately for each subpopulation.

The graphical representation of these block partitions for each data sample exhibits quite different patterns, so that, for example, the four gamete rule (shown as part (c) in all three Figures) tends to produce the most refined block structures and the confidence intervals' criterion (part (b)) allows for greatest breaks between blocks. Nevertheless, for each data sample the overall configuration of the haplotype structure can be distinctly identified for

the partitions supplied by Haploview (shown in parts (b)-(d)). It is also clear that the block partitions obtained from *HAPLOGEN* (part (a) in each figure) fit the overall configuration of block structures for each data sample. Thus, the results in block partition provided by *HAPLOGEN* in general agree with those provided by the most common methods.

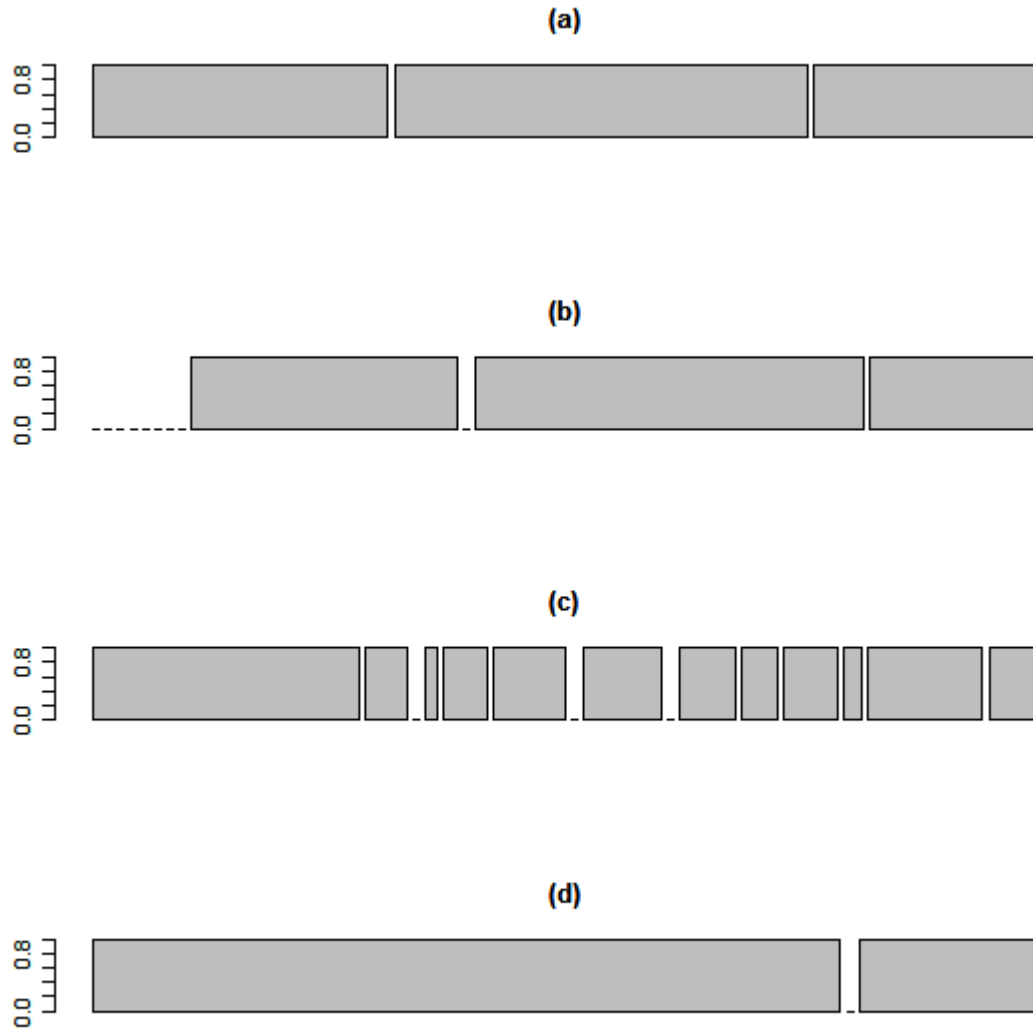


Figure 5.6 Block structure obtained for the JPT+CHB data from (a) *HAPLOGEN*, (b) confidence intervals, (c) four gamete rule and (d) solid spine of LD.

5.2.6 Summary of results for *HAPLOGEN* algorithm

Overall results indicate that the proposed genetic algorithm *HAPLOGEN* can be successfully applied to long sequences of SNP to obtain highly accurate haplotype resolution and an adequate block structure.

The algorithm uses two global optimization criteria reflecting the parsimonious principle: minimal number of distinct whole-length haplotype patterns (*min ds*) and the minimal number of total common haplotype patterns across all blocks (*min ncompat*). The analysis shows that the first criterion alone should be applied to the short sequences of SNP (around 60 positions or less) and the double criterion (*min ds & min ncompat*) is more suitable for the longer sequences. The proposed algorithm *HAPLOGEN* operates very fast: the running time is comparable to that of the fastest haplotype resolution algorithms (*fastPHASE*, *GERBIL*, *HAP*). For the most studied data samples the accuracy of prediction for the heterogeneous positions is within the range of the “block-based” group of algorithms (*GERBIL*, *HAP*, *HaploBlock*) for haplotype resolution. One particular data set (YRI HapMap data) exhibits relatively high switch rate compared to the other existing algorithms for haplotype inference. This may be due to the fact that the parsimony principle may not be enough to fully describe the real data. Since *HAPLOGEN* is able to find the minimal number of patterns (equal to the true number of patterns) within each block, while still retaining high errors, then it implies that the exact patterns found are not always the true ones and, therefore, other selection/optimization criteria need to be included into the model. In particular, this point is well illustrated by the fact that *HAPLOGEN* finds two solutions to the Drysdale data (see Section 5.2.1, Table 5.3) both with the same number of patterns, with only one solution being the true one. This problem needs to be addressed in the future to improve the accuracy of haplotype resolution. One possible approach may be using the perfect phylogeny within any particular block. The high error rates for the Yoruba population and low error rates for the European derived population can justify the perfect phylogeny approach: the way the genotype data was collected implies that the Yoruba sample comes from a very conservative population where historic mutations could be traced much more easily than in other populations with greater number of ancestral genotypes. This logic, therefore,

explains why the parsimonious principle is able, in fact, to work well for the European derived sample, while the Yoruba population may need to be described by perfect phylogeny.

As a part of the solution, *HAPLOGEN* obtains a block structure that is consistent with most other block partitioning methods. The proposed algorithm also provides a new feature that is not available in the existing haplotype resolution and block partition methods: the scores for block boundaries are obtained. These have proven useful in the prediction of possible block boundaries and in general indicate the strength of the left side of each position as a block boundary.

The proposed algorithm also successfully incorporates missing data into the model. The algorithm achieves reasonable accuracy in predicting missing data.

Chapter 6

Application of HAPLOGEN algorithm to population studies

The possibility of the existence of differences in haplotypes and haplotype block boundaries for different populations have long attracted the attention of researchers. The extent of the differences and similarities of the haplotype patterns and block structures in human populations must have had a tremendous impact on the construction of the haplotype map of human genome [102]. The fact that SNP frequencies differ among populations by about 15% [102] is attributable to the differences in haplotypes (as haplotype patterns) among populations. Goldstein and Weale in 2001 [103] determined that patterns of linkage disequilibrium can be quite different among populations. These findings imply that there must be differences in block structures among human populations. One of the first studies of the block structures in different populations was conducted by Gabriel *et al.* [15]. By characterizing the haplotype patterns spanning 13Mb of the human genome in samples from Africa, Europe and Asia, they determined the block structure for each of the populations and concluded that the boundaries of blocks as well as specific haplotypes within those boundaries are highly correlated across populations. Another study performed by Liu *et al.* [104] supported these findings only partially. The authors have applied the dynamic programming algorithm proposed by Zhang *et al.* [11] using the block definition of Patil *et al.* [2] with a threshold of 95% to the data from 16 worldwide populations on chromosome 10. It was shown that significant similarity of block boundaries exists within the European group of populations and also within part of the African group of populations. The difference in block structures was shown to be present within each group of east and north Asians, Americans and most of the Africans. The differences in the African populations can be explained by the fact that that continent has the longest history and the richest ethnic diversity. Groups differences among populations were also studied and the results confirmed the fact that most of the time the block structures have not exhibited similarity among populations from different geographic regions. There were, however, certain cases when the similarity was present, for example, between Biaka (Africa) and Irish (European) and between Yoruba (Africa) and Japanese (Asia). A possible explanation suggested by the authors [104] is that some African populations may have had European or Asian relations

back in their history. In general, the overall conclusion of the study was that the block boundaries are significantly different across populations. This contradicts the results previously provided by Gabriel et al. [15]. Gu *et al.* [105] have examined the genotype data in 38 populations and, while confirming previously found results of significant differences among worldwide populations, showed that there actually exist conservative tagSNP (representative SNP) patterns across populations.

Menashe *et al.* [106] found the differences in the collection of haplotypes common for groups of genotypes drawn from different populations. They analyzed the data for a 400 kb olfactory receptor (OR) gene cluster on human chromosome 17p13.3 obtained from 35 individuals. The individuals represented four different ethnogeographical groups: Pygmies, Bedouins, Yemenite Jews and Ashkenazi Jews. The genotype data of length 74 SNPs were subjected to the haplotype decomposition using a variation of Clark's algorithm, and the differences in haplotype patterns were then studied. Analysis of the distribution of specific haplotype patterns within each ethnic group revealed significant pairwise differences between these groups. The highest difference was observed between Pygmies and Ashkenazi Jews. Analysis of linkage disequilibrium within each group indicated considerable differences in the spatial distribution of LD across these four populations. The overall conclusion of this study is that there are significant differences in haplotype patterns (and their distributions) and the linkage disequilibrium among the four studied populations. The difference found within the OR gene suggests the functional difference of this gene in human populations across the world.

Most of the current studies support the theory that there are considerable differences in the variety of haplotype patterns as well as in the block boundaries among populations with different ethnogeographic origins. Therefore, the block structure and the haplotype decomposition obtained from mixed populations samples will not necessarily reflect the true solution or solutions. The problem that needs to be addressed is how to separate different populations in a sample and construct their respective solutions, i.e., block structure and the haplotype decomposition. While most of the time the ethnicity of an individual in a sample is known in advance, it may not always be fully informative since any particular individual may have a genetic relation to another population that is not immediately obvious or it may even represent a hidden subpopulation within the same group. In reality, any sample

representing some distinct ethnogeographic group always contains a fraction of genotypes that are the result of assimilation with some other population groups. Therefore, it is important to be able to detect individuals representing the “core” or “founder” haplotype patterns that are specific only for this particular population, and, on the other hand, to determine those individuals that may be a product of assimilation between different populations.

There are very few studies that have developed methods to infer haplotypes for different populations within a given sample and to obtain block structures for each such sub-population. One such study was done by E.P.Xing *et al.* [107]. The authors propose the algorithm for multi-population haplotype inference. The goal of their study was to jointly infer the haplotypes from a sample of genotypes in sub-populations represented in the sample. In each group (sub-population) the algorithm (called *HDP-Haplotyper*) finds the set of “founders” (haplotypes unique for this particular sub-population). The same algorithm also determines the set of haplotypes shared between sub-populations by a Bayesian approach to haplotype inference using the hierarchical Dirichlet process mixture [108]. *HDP-Haplotyper* was shown to perform better (in terms of accuracy of haplotype decomposition) than other haplotyping algorithms (*PHASE* [57], *DP-Haplotyper* [108] and *HAPLOTYPER* [58]). This algorithm showed very good accuracy and worked on more than two sub-populations, but was not designed to perform block partitioning within respective groups or to compare their block structures. Thus, *HDP-Haplotyper* is not suitable for the long sequences. The authors only used their algorithm for data no longer than 10 SNPs. In addition, the present version of *HDP-Haplotyper* does not perform clustering of the individuals since it is assumed that the population labels are known in advance, although it is mentioned in the paper that the clustering modification is straightforward.

A study done by G.Kimmel, R.Sharan and R.Shamir [9, 10] concentrated on identifying haplotype blocks for different populations from haplotype data. For a given set of haplotyped individuals, the algorithm partitions the sample into different populations and then searches for the block partitions within each population. The problem is formulated in the form of the Minimum Block Haplotypes (MBH) problem, where the result is achieved by minimizing the total number of distinct haplotypes across all sub-populations and their blocks. The block partitioning part of the algorithm is done using a dynamic programming

method similar to one proposed by Zhang [11]. The algorithm also incorporates missing data into the model. When tested on real and simulated data the algorithm performed very well. In the case of up to four sub-populations, it was able to correctly classify 70-99% of the haplotypes (results vary for different amounts of missing entries). When applied to genotype data for two sub-populations (where heterogeneous positions are treated as missing entries) the algorithm was able to correctly classify over 95% of the haplotypes. The high accuracy rate can partially be explained by the fact that the data used in testing was the collection of genotypes from parent-offspring trios which may not be assumed entirely independent. In addition, as mentioned by the authors [9, 10] the genotype data that they used contained a relatively low fraction of ambiguous sites. Thus, the algorithm proposed by Kimmel *et al.* [9, 10] showed very promising results; in particular, it had a very high rate of correct classification for two (when using genotypes) and up to four sub-populations (when using haplotypes) while incorporating the missing data. A limitation of this method is that it does not specifically account for the genotype data where heterogeneous positions carry information that may contribute to the improved accuracy. Therefore, population classification of the mixed genotype samples remains a computational challenge.

In contrast to the existing algorithms, the algorithm *HAPLOCLUST* suggested in this section has the following benefits:

- (a) works on genotypes with missing data and unknown population assignment;
- (b) given a genotype sample, extracts two clusters of genotypes with significantly different block structures and collections of haplotypes;
- (c) produces haplotype resolution of high accuracy within each group/cluster.

Thus, the algorithm *HAPLOCLUST* is suitable for the two-population haplotype inference and block partitioning for long SNP sequences. The proposed clustering algorithm is based on the genetic algorithm *HAPLOGEN* for haplotype resolution and block partitioning, which was shown to produce accurate haplotype inference and block structure comparable to that of the existing methods. The algorithm was implemented as an extension of the *HAPLOGEN* algorithm for haplotyping and block partitioning in the form of an optional “method” specification. That is, if the method is specified as “*CLUST*” in the genotype function call, the algorithm will find the solution for the mixed sample and for the two separate groups or clusters.

The input for the proposed algorithm *HAPLOCLUST* is a sample of genotypes assumed to be a mixture from two populations with unknown population labels. The algorithm performs the following two tasks. In the given genotype sample, it identifies two sub-populations that differ substantially from each other in their haplotype block structure and, as an intermediate result, in their collection of haplotype patterns; for each such group (or cluster) the algorithm constructs the full haplotype and block structure profile. A portion of the individuals may be left unclustered. These genotypes refer to the individuals that are difficult to assign to either of the extracted clusters. They may be interpreted as either the result of assimilation of the two sub-populations or simply not belonging to any of them. The outline of the algorithm is given in Fig. 6.1 and is discussed below in detail.

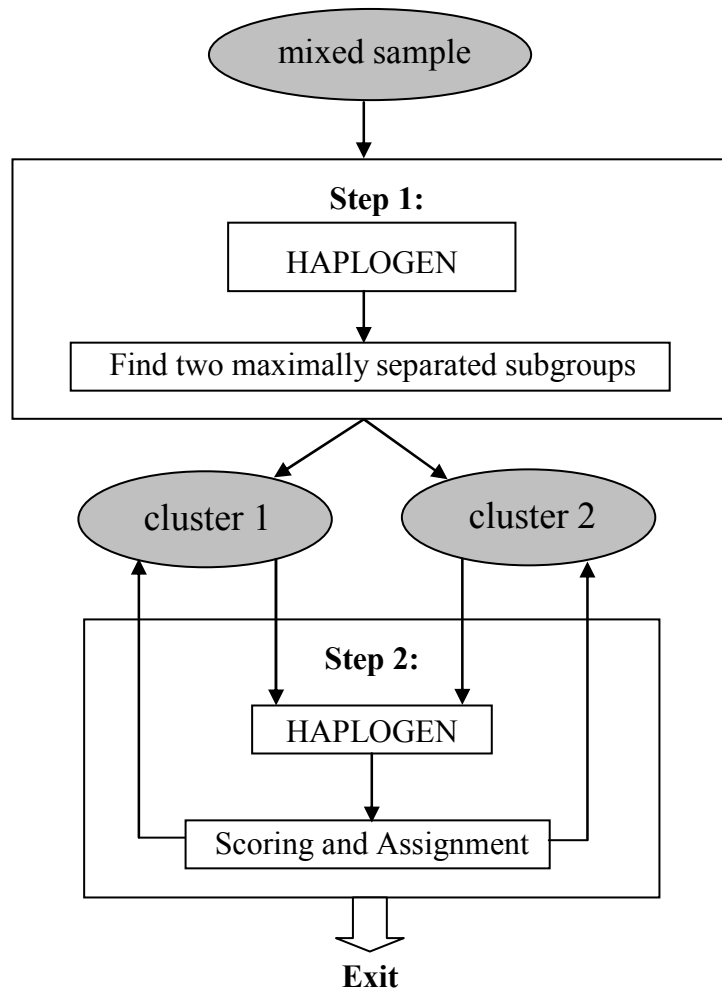


Figure 6.1 Outline of the *HAPLOCLUST* algorithm.

6.1 Clustering algorithm: step 1

The first step of the algorithm *HAPLOCLUST* relies on the fact, that in the block structure obtained from a mixed sample, some block boundaries of the found ones will not be present (will be skipped) for certain subgroups of that sample. Therefore, the goal of the first step of the algorithm is to determine two such block boundaries that would correspond to the two largest maximally separated subgroups. The maximality refers to the total number of individuals skipping one such boundary but not the other. The found subgroups are meant to represent core genotypes for the two sought clusters. In search for these two boundaries, the genetic algorithm *HAPLOGEN* for the haplotyping and block partitioning is run on the entire mixed sample. In the obtained block structure for these data, every two adjacent blocks are considered as one potential block for a certain subset of the sample, skipping the boundary between two such blocks. Every such potential block, thus, represents that particular block boundary. Within a potential block the distinct haplotypes are obtained based on the available haplotype matrix. Among the distinct haplotypes the two most common haplotypes are determined. After that the genotypes that could be resolved using any of these common haplotypes within the given potential block are identified (marked). For every potential block this information can be represented as a (0,1)-vector of length n , where 0 at position i indicates that the i -th genotypes cannot be resolved using any of the common haplotypes determined within this block, and 1 indicates otherwise. After the $|\mathbf{B}|-1$ potential blocks (where $|\mathbf{B}|$ is the number of blocks in the block structure for the entire sample) are examined, $|\mathbf{B}|-1$ such vectors of length n are obtained. For every pair of these vectors \mathbf{v}_j and \mathbf{v}_k , the following two quantities are calculated:

$$d_{jk} = (\mathbf{v}_j, \mathbf{v}_k) = \sum_{i=1}^n v_{ij} \wedge v_{ik} = \sum_{i=1}^n v_{ij} v_{ik} \quad (6.1)$$

$$c_{jk} = \sum_{i=1}^n v_{ij} \vee v_{ik} \quad (6.2)$$

The first quantity d_{jk} is based on binary disjunction and is a scalar product which represents the number of overlapping genotypes in the two vectors (potential blocks). The value of d_{jk} indicates how many genotypes share the common patterns for the two potential blocks which is the same as the number of genotypes skipping both block boundaries corresponding to the two potential blocks. The second quantity c_{jk} is based on the binary

conjunction operation and represents the overall spread of the joined genotypes from the two vectors. Thus, the value c_{jk} indicates the overall coverage of the genotypes by the two sets of common patterns from the two potential blocks. This is the same as the number of genotypes skipping either of the block boundaries corresponding to the two potential blocks.

For example, the two quantities are computed for the following two vectors

v_i : 0 0 1 1 1 1 1 1 1 0 0 0 0 0

v_k : 0 0 0 0 0 1 1 1 1 1 1 0 0 0

are: $d_{jk} = 4$ and $c_{jk} = 9$.

In search of the two subgroups of genotypes with the maximally different block structure, it is necessary to find the two block boundaries that are not mutually shared by the representatives of these two groups. At the same time, the number of genotypes resolvable by the corresponding common haplotypes should be as large as possible.

Therefore, in order to find two maximally separated groups of genotypes, one needs to determine the two vectors with the minimal d_{jk} and the maximal c_{jk} . It should be noted that simply maximizing the difference ($c_{jk} - d_{jk}$) does not work well enough. The two subgroups should also create the largest set, i.e., the value of c_{jk} itself should also be maximal. The vectors v_i and v_k represent the two boundaries being sought. Then, using the same vectors, all genotypes are assigned to group one or group two or left unclassified according to the following principle: if the genotype is marked by 1 in vector v_i but not v_k it is assigned to group 1; similarly if it is marked by 1 in v_k but not v_i , it is assigned to group 2. If a genotype is marked by 0's or 1's in both vectors, it is considered to be unclassified.

6.2 Clustering algorithm: step 2

While step one determines the core for the two clusters, by finding the two largest subgroups with maximally separated block structures, the second step iteratively uses the information from the current content of each cluster to include more genotypes from the rest of the sample. As the first stage of each iteration, two separate block structures (together with haplotype decompositions) are determined for the two extracted clusters of genotypes by applying the genetic algorithm *HAPLOGEN* for haplotyping and block partitioning. Next, the unclustered genotypes are rated as to which group each one of them belongs. The rating is performed by using two cluster scores for every unclassified whole-length genotype. The

cluster score of a genotype with respect to either cluster is calculated as a proportion of the length of the genotype, which can be completely resolved using available haplotype patterns from the haplotype decompositions within blocks provided by that cluster's profile. This process starts by considering one block after another and setting the initial value of the cluster score to 0. If a genotype can be resolved by the two complementary haplotypes, both represented in the current block, the score is increased by the size of this block. Otherwise the score is not increased and the process moves on to the next block. There are two scores (one from each cluster) obtained for each genotype. A genotype is assigned to the cluster whose respective score is the highest for this genotype. If the two scores are the same, the genotype remains unclustered. If any of the two groups is expanded as a result of such rating and classification, the iteration is repeated: the genetic algorithm is applied to the updated clusters and the remaining genotypes are attempted to be clustered.

This process continues until the assignment can no longer be determined or the entire sample is completely separated into two clusters. The algorithm *HAPLOCLUST* is not guaranteed to separate all genotypes into two clusters; a small part of the sample may be left unclustered. These are the candidates for another cluster or the results of the assimilation of the two sub-populations.

It is worth noting that even though the proposed algorithm starts out by finding the two subgroups with significantly different block structures (as implemented by step 1). The second part of the algorithm (step 2) is designed to assign the genotype to either cluster using the similarity of the haplotype decompositions. Thus, the resulting clusters differ not only in their block structures but also in the collection of short and long-range haplotypes.

6.3 Testing the clustering algorithm

6.3.1 Simulated ACE data

The proposed algorithm *HAPLOCLUST* was tested on the ACE data [98] in the following way. Originally, the ACE data represents two small subsamples taken from the Euporean (six genotypes) and African (five genotypes) populations. The original sample does not contain enough information to perform meaningful clustering and a simulation was therefore undertaken. According to [98], the six individuals from the European population produced four haplotypes (each of length 52 SNPs), and the five individuals from the

African population produced ten haplotypes. The European subsample was generated by randomly pairing the four basic genotypes to obtain 20 genotypes, and the same was done for the African subsample to get 50 genotypes. The ten basic haplotypes for the African population contained one haplotype that was also listed among four basic haplotypes for the European population. The first simulation (simulated ACE data I) used four haplotypes for generation of the European subsample and all ten haplotypes for the African subsample. The second simulation (simulated ACE data II) used only non-intersecting groups of haplotypes, i.e., four haplotypes for the European subsample and only nine haplotypes (one haplotype overlapping with the European group was excluded) for the African subsample. Results from running *HAPLOCLUST* on the two simulated data sets are represented in Tables 6.1, 6.2 and 6.3.

Table 6.1. Results from applying *HAPLOCLUST* to simulated ACE data I

	Computed cluster 1	Computed cluster 2	Unclustered
True European	0	6	13
True African	39	2	9

Table 6.2. Results from applying *HAPLOCLUST* to simulated ACE data II

	Computed cluster 1	Computed cluster 2	Unclustered
True European	0	19	1
True African	48	2	0

Analysis on the block structure obtained from applying the *HAPLOCLUST* algorithm for the ACE data I shows that the mixed data exhibits considerably more refined block structure than the two extracted clusters: the mixed sample is partitioned into the five haplotype blocks, spanning positions 1 – 7, 8 – 23, 24 – 40, 41 – 47, and 48 – 52, while the two clusters have three and one block spanning positions 1 – 30 , 31 – 43 , 44 – 52, and 1– 52, respectively.

The results indicated in Table 6.1 can be explained by the following logic. The reason that there is a number of genotypes with undetermined cluster assignment is due to the overlapping collection of haplotypes used to generate the two subgroups. More precisely, there is one particular haplotype that appears in both groups. When this haplotype is involved in any genotype it creates confusion for the algorithm as to which cluster the genotype should be assigned to. The results from applying *HAPLOCLUST* to the simulated

ACE data II show a clear distinction between true populations (see Table 6.2). In particular, there is a distinct correspondence between the first cluster and the generated true African group of genotypes, and between the second cluster and the generated true European group of genotypes. The accuracy of classification for the ACE data II was 95% for the European and 96% for the African simulated samples with an average of 95.7% in overall prediction.

According to the *HAPLOCLUST* output, the mixed simulated ACE data II data has block structure similar to the one from the mixed simulated ACE data I, also resulting in the five blocks identified at 1 – 8, 9 – 14, 15 – 40, 41 – 48, and 49 – 52. The two clusters (“African” and “European”) produced by the algorithm turned out to have trivial block structure: every cluster is represented by a single block spanning positions 1 – 52. This result is not surprising, but rather an indication of a clear separation since the original (true) African group was constructed by using only nine haplotypes and the original European group by using only four haplotypes. The initially limited number of haplotypes, spanning any particular sample, essentially implies trivial block structure for that sample.

The analysis performed on the two sets of simulated data shows that, when the true haplotypes corresponding the two groups are non-overlapping, the proposed algorithm *HAPLOCLUST* is able to accurately separate two clusters of genotypes according to different block structure and the collections of haplotypes.

6.3.2 Data from four populations

Data from four populations (Pygmies, Bedouins, Yemenite Jews and Ashkenazi Jews) was originally provided in a study by Menashe *et al.* [106]. They contain the genotyped data for a 400 kb olfactory receptor (OR) gene cluster on human chromosome 17p13.3 obtained from 35 individuals (7 genotypes of each Pygmy, Bedouin and Yemenite Jew group and 9 genotypes from Ashkenazi Jews). Even though each group’s sample size is not large the data were nevertheless analyzed using the *HAPLOCLUST* algorithm. The genotype data contains 74 SNPs, out of which 40 had rare allele frequency equal to 0.15 or higher. Out of these 40 sites, 37 had alleles that were not intact or were disrupted and were, therefore, selected for the analysis.

Results from running the *HAPLOCLUST* algorithm on the 6 pairs of different population samples are shown in Table 6.3. Every row in the table represents output for some particular pair of samples. Cell entries indicate identification labels for the genotypes

from the sample marked by the column. Thus, for example, in pair 5, 3 out of 7 Bedouin individuals were assigned the group 2 label, 1 was assigned the group 1 label and 3 left unclassified (labeled 0); among the Pygmies individuals, 4 were assigned to group 1, 1 individual assigned to group 2, and 2 left unclassified. In this particular pair, it is therefore reasonable to assume that group 1 can be associated with Pygmies and group 2 with Bedouins.

Table 6.3 Results from running *CLUST* algorithm on the pairs of samples.

	Bedouins	Ashkenazi jews	Yemenite Jews	Pygmies
Pair 1 →	0 0 0 0 1 0 0	2 0 0 0 0 0 2 0		
Pair 2 →	0 0 0 0 0 0 1		0 2 2 2 2 0 1	
Pair 3 →			1 2 2 2 0 0 2	0 0 1 0 0 0 0
Pair 4 →		0 0 2 0 0 0 0 0 0		0 0 0 0 0 1 1
Pair 5 →	2 0 2 2 0 0 1			2 0 0 1 1 1 1
Pair 6 →		1 1 2 1 1 1 1 1 1	1 2 2 1 2 1 1	

The overall conclusion in the four population data samples is that the algorithm is able to correctly classify some portion of the individuals in almost every pair, with a visible distinction between groups. A large fraction of unclassified individuals can be attributed to the insufficient information (small sample sizes of the population groups and relatively short SNP sequence under study) and to the similarity of the haplotype patterns due to possible assimilation as, for example, in the case of Ashkenazi and Yemenite Jews.

6.3.3 HapMap data

The next data to be tested using the *HAPLOCLUST* algorithm was the same HapMap data used by the *HAPLOGEN* algorithm in section 5.2.5. Namely, the *HAPLOCLUST* algorithm was applied to the data from four different world populations CEU (European decent, 30 genotypes), YRI (Yoruba, 30 genotypes), CHB (Han Chinese, 45 genotypes) and JPT (Japanese, 45 genotypes). The data from the four samples were taken from a randomly chosen chromosome (chromosome 2) in the release 22 in HapMap web site. In order to provide a reliable result, only children from 30 family trios from each CEU and YRI sample were considered. The genotype information from the parents was not used for the determination of the resolvable heterogenous positions in children genotypes. Thus, all four samples were original genotypic data from completely unrelated individuals. The 500 SNPs

having the same location (the first 500 positions of the data) along the chromosome were processed by the clustering algorithm.

Table 6.4. Group assignment of genotypes in different pairs of data sets of unrelated individuals

Data set	Cluster assignment	Accuracy
CEU + YRI	CEU: 11111111111121111211112111111111 YRI: 22222222222222222222222222222222	0.9 0.967
CHB + JPT	CHB: 2220000000200000121000000022002021020 00000020 JPT: 0002202200202200202000012022121002022 20102002	0.24 or 0.07 0.09 or 0.42
CEU + CHB	CEU: 22222222222222222222222222222222 CHB: 12111121111111111111111111111111121111 11111111	1.0 0.93
CEU + JPT	CEU: 22222222222222222222222222222222 JPT: 111111111211111111111111121112112111111 11111111	1.0 0.91
YRI + CHB	YRI: 11111111111111111111111111111111 CHB: 2220002100202200202000002022222022220 00000200	1.0 0.44
YRI + JPT	YRI: 22222222222222222222222222222222 JPT: 111111111111111111111111121112112111211 11111111	1.0 0.91

The results from running the algorithm on the pairs of data samples are shown in Table 6.4 in the form of the cluster vector produced by the algorithm. As mentioned above, the cluster assignment can either be “1”, “2” or “0” (unclustered). The accuracy of the assignment was determined on the basis that most genotypes were labeled by the same symbol (“1” or “2”) in either of the samples. Thus, the true assignment in most cases is visible through the difference in distribution of labels in the two samples.

The overall results are very encouraging in the sense that the proposed clustering algorithm was able to classify the populations with a high degree of accuracy; in the majority of cases the accuracy exceeded 0.91 in every pair. The only pair of samples that clearly stands out is the CHB+JPT data, where it wasn’t possible to classify populations into two distinct clusters with significantly different haplotype block structures. This can be explained by the relative similarity of genotypes as well as the block structures within this particular SNP region among Asian populations due to their common ethnogeographic ancestry. The analysis of some other SNP regions might reveal more distinction between these two populations. The results from the rest of the data allow us to conclude that the

genotypes of the populations from very distant parts of the world do, in fact, differ in their block structure and the haplotype distribution.

The difference in block structures for the pairs of populations can be traced by the scores for block boundaries as a part of the solution provided by the algorithm *HAPLOCLUST* (and, in particular, the *HAPLOGEN* part of it). The scores for block boundaries indicate how likely it is for the particular position to have a block boundary immediately to the left of it. These scores were obtained from the pooled sample and separately from each of the extracted clusters. An example of such an analysis performed on YRI+JPT data is shown in Fig. 6.2. The top graph clearly shows the mixture of the two different “signals,” which are shown on the two graphs below the top one.

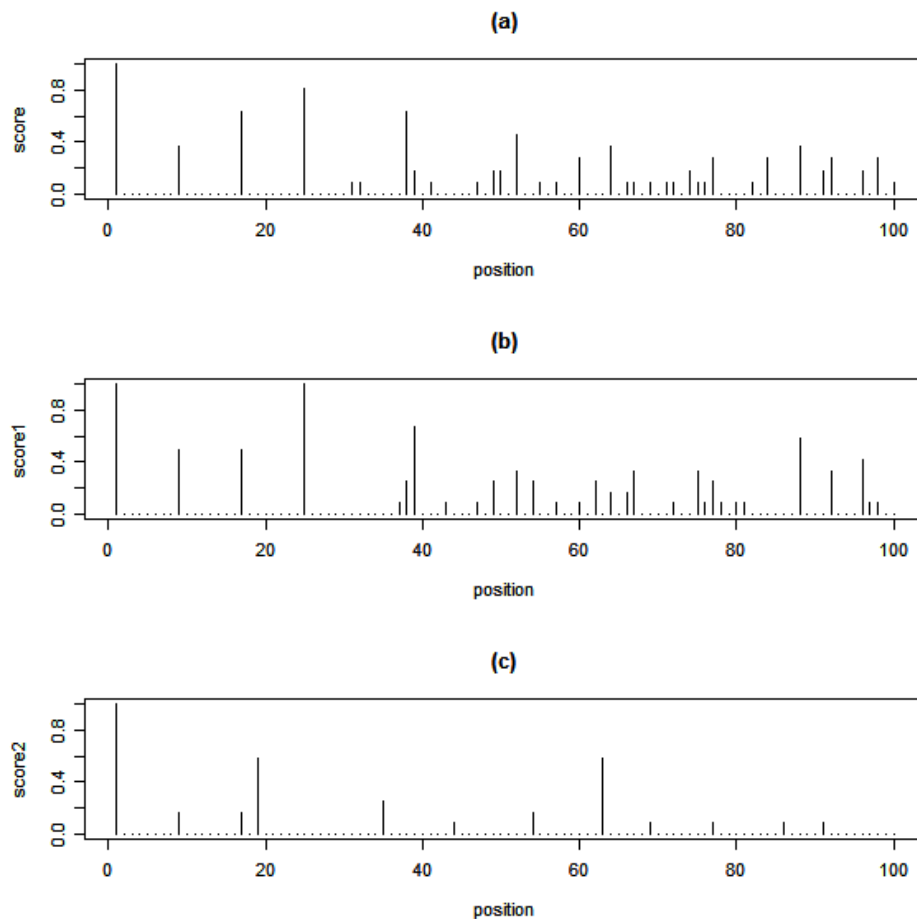


Figure 6.2. Scores for block boundaries from (a) the pooled data (YRI+JPT); and the data from each of the clusters inferred by the *HAPLOCLUST* algorithm: (b) cluster 1, (c) cluster 2.

Similarly, one can look at the actual block boundaries produced by the algorithm for the pooled sample and the two clusters shown in Fig. 6.2. The two block partitions (b) and (c) in Fig. 6.2 of the two clusters produced by the algorithm clearly exhibit quite different block boundaries. Compared to these two partitions, the pooled sample's block structure (shown in part (a) of Fig. 6.3) is very uninformative as it tends to have the block of almost equal length. Thus, the classification of a genotype sample into the clusters with different block structures helps to refine the block structure of the mixed sample.

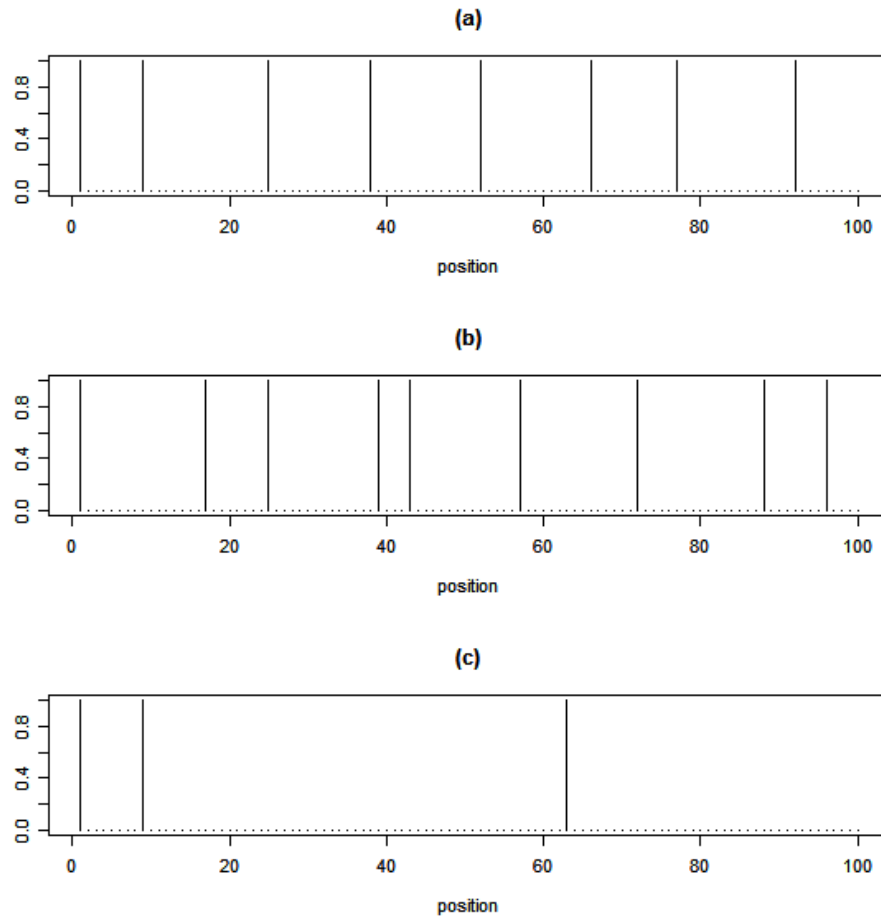


Figure 6.3. Actual haplotype block partitions obtained for (a) the pooled sample (YRI+JPT); and separately for each of the inferred clusters: (b) cluster 1, (c) cluster 2.

6.4. Summary of the clustering algorithm results

The proposed algorithm *HAPLOCLUST* is designed to cluster the given genotype sample into two groups with maximally different block structures when the population

assignment is not known or is suspected to be inaccurate. In addition to cluster extraction, the proposed algorithm produces the haplotype resolution and the full block structure profile for each cluster of genotypes. Analysis of the simulated data on angiotensin converting enzyme (ACE data) shows that the proposed algorithm can successfully separate two sub-populations with non-overlapping sets of haplotype patterns. In the presence of overlapping haplotype patterns the clustering is not always consistent with the true population assignment. The analysis of larger data sets (four paired samples from European, Asian and African populations obtained from HapMap) of unrelated individuals confirms the high average rate of correct classification of around 95%. Previous methods [9, 10] showed similar classification rate but, unlike ours, was performed on the sets of partially related individuals which could implicitly affect the correctness of results. In addition, the previous methods were not specifically designed for treatment of genotype data: the genotype data were used as haplotype data with missing entries at ambiguous positions. Despite good results shown on the data with a relatively low proportion of ambiguous sites, these methods can be challenged when the data contains a large proportion of ambiguous positions.

Our findings of impossibility of clustering of the pooled Asian sample (Japanese and Han Chinese) suggest that these two populations may have common ancestry which is reflected in the similarity of common long-range haplotypes and the block structure. This conclusion, however, may only be applied to the particular part of chromosome 2 that we studied. More precisely, the two Asian populations may exhibit very different block structures in other genomic regions. The fact that some of the genotypes can be left unclassified implies that the proposed algorithm is able to extract only the clusters with substantially different block structures and collections of haplotypes. Too many unclustered genotypes may indicate the presence of assimilation in the sample. The proposed clustering algorithm was shown to be very fast: the processing time for the samples of 60 genotypes of length 500 SNPs took 15-20 minutes on a 2.3 GHz processor.

Chapter 7

Conclusions and future work

Discovery of the variations in the human genome characterized by single nucleotide polymorphisms (SNPs) opened an entirely new area of research in bioinformatics. In particular, SNPs have proven to be useful in gene mapping for complex human diseases. The fact, that complex traits (expressed as complex diseases and other phenotypic characteristics) have compound genetic component has increased the need for the availability of the haplotype decomposition for the full-length of the specific regions of interest. The existence of the haplotype blocks as regions of limited haplotype diversity and possibly high linkage disequilibrium (LD) provided additional breakthroughs in the analysis of complex traits. Thus, the reliable methods of haplotype resolution and block partitioning are needed to facilitate the effective genetic mapping of complex traits. Since available experimental haplotyping technologies are often expensive, computational methods offer a good alternative. In particular, the methods of simultaneous haplotyping and block partitioning are on the edge of the current research in this area. Although there are currently several methods that provide good results to that problem, there is still room for improvement.

In this dissertation, a new algorithm *HAPLOGEN* that simultaneously finds haplotype resolution and block partitioning is proposed. The algorithm directly incorporates missing data so that the entries to those positions comply with the entire model and are found during the process and not as a separate step. This feature is not always available in existing algorithms for haplotype resolution and block partitioning. The algorithm also features such advantages as the usage of the multilocus LD criterion (Normalized Entropy Difference) for block partitioning and unlimited length for haplotype blocks. The use of the LD-based multilocus criteria for the purpose of haplotype block identification corresponds to mainstream thoughts on haplotype block definition. In addition, the proposed algorithm calculates the scores of block boundaries at each SNP position. Each score approximately indicates how likely it is for this particular SNP to have a block boundary immediately to the left of it. Although previous studies on block partitioning produced the measures of degree of strength for block boundaries [14], the genetic algorithm proposed in this dissertation

offers the possibility of finding block partitioning together with haplotype resolution while providing the scores for the block boundaries. The results from running the algorithm on real and simulated data, including the HapMap data, have shown that the *HAPLOGEN* algorithm provides accuracy comparable to that of the existing state-of-the-art methods for most studied samples. There are minor adjustments that need be done in the future to improve the accuracy for certain data. The proposed algorithm achieves the same running time as the other best algorithms for haplotype resolution and block partitioning. The block boundaries output by the *HAPLOGEN* algorithm in general agree with those of the other algorithms. These successful results generally support the parsimonious principle used as an assumption for the haplotyping problem and incorporated into the model in the forms of the fitness function and global optimization criteria.

The proposed algorithm can process relatively large data (up to about 350000 entries of the genotype matrix) but has problems accepting larger data sets. This limitation is not set by the algorithm itself but imposed by the R software capabilities. Also, improvements may be done in the block partitioning part of the algorithm, which includes investigation of the ways to use only one block identification criterion (namely, multilocus LD measure, NED) without posing any restrictions on the block size.

The run time of the proposed algorithm *HAPLOGEN* is approximately equal to that of the fastest existing methods for haplotype resolution and block partitioning. The overall time complexity of the algorithm is $O(n^3m^3)$, if the number of iterations is set to the default value.

Despite overall good results provided by algorithm *HAPLOGEN*, there are several ways it can be improved. First, the fact that some data exhibited higher error rates indicates that the parsimonious principle is not able to fully describe the real data in all cases, which is especially noticeable in data from conserved populations with a small number of ancestral haplotypes. This stipulates the need to include additional global optimization or local selection criteria. The most appropriate approach seems to be perfect phylogeny within any particular block at each iteration. Several previous studies [59, 62, 63, 64] showed that perfect phylogeny is a reasonable assumption for short sequences of SNP that can provide additional insight into the nature of haplotype resolution. The inclusion of the additional criteria into the model should ensure lower error rates for all data. Intermediate results show

that incorporating if the second-order Markov chain into the mutation step may significantly decrease the error rate.

Second, the accuracy of haplotype decomposition can be slightly increased by improving the haplotype resolution within each block, namely, by performing several iterations of the algorithm locally within each block as part of the global iteration for the whole-length sequence. This may minimize the occurrence of badly decomposed blocks as a part of the overall solution and will lead to the faster achievement of the global optimization criteria.

Third, ways to improve the applicability of the multi-locus linkage disequilibrium measure (given by NED) to long sequences of SNP should be investigated. As of this moment, NED is only capable of detecting linkage disequilibrium in sequences no longer than 10 SNPs, while blocks can be as long as 100 SNPs. Due to this fact, additional block identification criteria is currently used in the *HAPLOGEN* algorithm. The extension of the use of NED measure for long sequences of SNPs should improve the reliability of the block boundaries.

Fourth, the block boundaries scores may need to be incorporated into the final solution (block structure) by using the threshold for each score and, thus, determining if a particular boundary qualifies to be represented in the final block structure.

Fifth, the confidence bounds on the number of iterations may be suggested rather than the single default value calculated from the linear regression (see Sec. 3.10).

Sixth, the prediction of missing data needs to be compared to that provided by the other algorithms for haplotype resolution to better assess the advantage of incorporating missing data into the model provided by the *HAPLOGEN* algorithm.

Ways to apply the proposed *HAPLOGEN* algorithms to the population studies were also explored in this dissertation. The extension of the *HAPLOGEN* in a form of the new algorithm *HAPLOCLUST* performs clustering of the given sample of genotypes into two groups with different block structures when the population assignment is not known in advance, and also finds the haplotype resolution and block partition for these two clusters. The *HAPLOCLUST* algorithm has obtained very promising results for real and simulated data. The results demonstrate the algorithm's ability to differentiate between two populations with a high degree of accuracy when there is little or no assimilation present.

As was shown by real genotype data, the *HAPLOCLUST* algorithm performs extremely well for the two-population mixed sample. However, the case of three or more populations should also be investigated and, thus, remains the goal of future research. Some insight for solving this problem can be learned from traditional clustering algorithms.

Bibliography

- [1] B.S. Shastri. SNP alleles in human disease and evolution. *Journal of Human Genetics*, 47: 561-566, 2002.
- [2] N. Patil, A.J. Berno, D.A. Hinds, W.A. Barret, J.M. Doshi, C.R. Hacker, C.R. Kautzer, D.H. Lee, C. Marjoribanks, D.P. McDonough, B.T. Nguyen, M.C. Norris, J.B. Sheehan, N. Shen, D. Stern, R.P. Stokowski, D.J. Thomas, M.O. Trulson, K.R. Vyas, K.A. Frazer, S.P. Fodor, and D.R. Cox. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, 294(5547):1719-1723, 2001.
- [3] M. Nothnagel. The definition of multilocus haplotype blocks and common diseases. Dissertation. Humboldt-Universität zu Berlin, 2004.
- [4] B.S. Shastri. SNPs and haplotypes: Genetic markers for disease and drug response (Review). *International Journal of Molecular Medicine*, 11: 379-382, 2003.
- [5] X.-S. Zhang, R.-S. Wang, L.-Y. Wu, L. Chen. Models and Algorithms for Haplotyping problem. *Current Bioinformatics*, 1: 105-114, 2006.
- [6] Daly M.J., Rioux J.D., Schaffner S.F., Hudson T.J., Lander E.S. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29: 229-232, 2001.
- [7] M. Nothnagel, R. Fürst, K. Ronde. Entropy as a measure for linkage disequilibrium over multilocus haplotype blocks. *Human Heredity*, 54(4): 186-198, 2002.
- [8] L. Moran. Random Genetic Drift. 22 January, 1993, *The Talk Origins Archive*. Accessed on 15 May, 2007 at <<http://www.talkorigins.org/faqs/genetic-drift.html>>
- [9] G. Kimmel, R. Sharan, R. Shamir. Identifying blocks and sub-populations in noisy SNP data. In *Proc. of the Workshop on Algorithms in Bioinformatics (WABI 2003)*: 303—319, 2003.
- [10] G. Kimmel, R. Sharan, R. Shamir. Computational problems in noisy SNP and haplotype analysis: block scores, block identification and population stratification. *INFORMS Journal on Computing*, 16(4): 360-370, 2004.
- [11] K. Zhang, M. Deng, T. Chen, Michael S. Waterman, and F. Sun. A dynamic programming algorithm for haplotype block partitioning. In *Proc. of the National Academy of Sciences USA*, 99(11):7335-7339, 2002.
- [12] N. Wang, J. Akey, K. Zhang, R. Chakraborty, L. Jin. Distribution of recombination crossovers and the origin of haplotype blocks: the interplay of population history, recombination, and mutation. *American Journal of Human Genetics*, 71: 1227–1234, 2002.
- [13] G. Greenspan, D. Geiger. Model-based inference of haplotype block variation. In *Proceedings of the Seventh Annual International Conference on Computational Molecular Biology (RECOMB 2003)*: 131–137, 2003.
- [14] M. Koivisto, M. Perola, T. Varilo, W. Hennah, J. Ekelund, M. Lukk, L. Penttonen, E. Ukkonen, H. Mannila. An MDL method for finding haplotype blocks and for estimating the strength of haplotype block boundaries. *Pacific Symposium on Biocomputing*: 502-513, 2003.
- [15] S. B. Gabriel, S. F. Schaffner, H. Nguyen, J. M. Moore, J. Roy, B. Blumenstiel, J. Higgins, M. DeFelice, A. Lochner, M. Faggart, S. N. Liu-Cordero, C. Rotimi, A. Adeyemo, R. Cooper, R. Ward, E. S. Lander, M. J. Daly, D. Altshuler. The structure of haplotype blocks in the human genome. *Science*, 296: 2225-2229, 2002.

- [16] J.D. Wall, J.K. Pritchard. Assessing the performance of the haplotype block model of linkage disequilibrium. *American Journal of Human Genetics*, 73: 502–515, 2003.
- [17] M.H. Stumpf. Haplotype diversity and the block structure of linkage disequilibrium. *Trends in Genetics*, 18(5): 226–228, 2002.
- [18] N. A. Rana, N. D. Ebenezer, A. R. Webster, A. R. Linares, D. B. Whitehouse, S. Povey, A. J. Hardcastle. Recombination hotspots and block structure of linkage disequilibrium in the human genome exemplified by detailed analysis of PGM1 on 1p31. *Human Molecular Genetics*, 13(24): 3089–3102, 2004.
- [19] D.B. Goldstein. Islands of linkage disequilibrium. *Nature Genetics*, 29: 109 – 111, 2001.
- [20] L.R. Cardon, G.R. Abecasis. Using haplotype blocks to map human complex trait loci. *Trends in Genetics*, 19: 135–140, 2003.
- [21] D. E. Reich, M. Cargill, S. Bolk, J. Ireland, P. C. Sabeti, D. J. Richter, T. Lavery, R. Kouyoumjian, S. F. Farhadian, R. Ward, E. S. Lander. Linkage disequilibrium in the human genome. *Nature*, 411: 199–204, 2001.
- [22] R. Judson, J.C. Stephens. Notes from the SNP vs. haplotype front. *Pharmacogenomics*, 2(1): 7-10, 2001.
- [23] Mendelian Inheritance. *Wikipedia*. Accessed on 17 May 2007 at <<http://en.wikipedia.org/wiki/>>
- [24] R. Mayeux. Mapping the new frontier: complex genetic disorders. *The Journal of Clinical Investigation*, 115(6): 1404–1407, 2005.
- [25] M. Nothnagel, J. Ott. Statistical gene mapping of traits in humans - hypertension as a complex trait: Is it amenable to genetic analysis? *Seminars in Nephrology*, 22(2): 105-114, 2002.
- [26] C. Meisel, T. Gerloff, J. Kirchheiner, P. M. Mrozikiewicz, P. Niewinski, J. Brockmoller, I. Roots. Implications of pharmacogenetics for individualizing drug treatment and for study design. *Journal of Molecular Medicine*, 81(3):154–167, 2003.
- [27] J.C. Stephens. Single-nucleotide polymorphisms, haplotypes, and their relevance to pharmacogenetics. *Molecular Diagnosis*, 4(4): 309-317, 1999.
- [28] J.K. Haseman, R.C. Elston. The investigation of linkage between a quantitative trait and a marker locus. , 2(1): 3–19, 1972.
- [29] R.C.Elston. Linkage and association. *Genetic Epidemiology*, 15(6): 565–576, 1998.
- [30] E.S. Lander, N.J. Schork. Genetic dissection of complex traits. *Science*, 265(5181): 2037–2048, 1994.
- [31] J. Ott,. Methods of analysis and resources available for genetic trait mapping. *The Journal of Heredity*, 90: 68-70, 1999.
- [32] D. Gordon, S.J. Finch. Factors affecting statistical power in the detection of genetic association. *The Journal of Clinical Investigation*, 115(6): 1408–1418, 2005.
- [33] L. Kruglyak, M. J. Daly, M. P. Reeve-Daly, and E. S. Lander. Parametric and nonparametric linkage analysis: a unified multipoint approach. *American Journal of Human Genetics*, 58(6):1347–1363, 1996.
- [34] C. Garner, L. Alison McInnes, S. K. Service, M. Spesny, E. Fournier, P. Leon, N. B. Freimer. Linkage analysis of a complex pedigree with severe bipolar disorder, using a Markov Chain Monte Carlo method. *American Journal of Human Genetics*, 68(4): 1061-1064, 2001.

- [35] J.Ott, J. Hoh. Statistical approaches to gene mapping. *American Journal of Human Genetics*, 67: 289–294, 2000.
- [36] L.R. Cardon, J.I. Bell. Association study designs for complex diseases. *Nature Review Genetics*, 2(2): 91–99, 2001.
- [37] K. Zhang, P. Calabrese, M. Nordborg, F. Sun. Haplotype block structure and its applications to association studies: power and study designs. *American Journal of Human Genetics*, 71:1386-1394, 2002.
- [38] N.E. Morton. Linkage disequilibrium maps and association mapping. *The Journal of Clinical Investigation*, 115: 1425–1430, 2005.
- [39] Y. Higashi, H. Higuchi, T. Kido, H. Matsumine, M. Baba, T. Morimoto, M. Muramatsu. SNP Analysis system for detecting complex disease associated sites. In *Proc. of the IEEE Computer Society Conference on Bioinformatics*: 450, 2003.
- [40] A. Collins, S. Ennis, P. Taillon-Miller, P. Y. Kwok, and N. E. Morton. Allelic association with SNPs: metrics, populations, and the linkage disequilibrium map. *Human Mutation*, 17(4): 255–262, 2001.
- [41] L.B. Jorde. Linkage disequilibrium and the search for complex disease genes. *Genome Research*, 10(10): 1435–1444, 2000.
- [42] M. Xiong, J. Zhao, E. Boerwinkle. Haplotype block linkage disequilibrium mapping. *Frontiers in bioscience*, 8: 85-93, 2003.
- [43] E.J.C.G. van den Oord, B.M. Neale. Will haplotype maps be useful for finding genes? *Molecular Psychiatry*, 9(3): 227-236, 2004.
- [44] H.T.Toivonen, P.Onkamo, K.Vasko, V.Ollikainen, P. Sevon, H. Mannila, J. Kere. Gene mapping by haplotype pattern mining. *IEEE International Conference on Bioinformatics and Biomedical Engineering*: 99 – 108, 2000.
- [45] H.T.Toivonen, P.Onkamo, K.Vasko, V.Ollikainen, P. Sevon, H. Mannila, J. Kere. Data mining applied to linkage disequilibrium mapping. *American Journal of Human Genetics*. 67(1): 133-145, 2000.
- [46] F.R. Zirwes. SNP Genotyping with single molecule fluorescence. *PharmaGenomics*, 3: 46-48, 2003.
- [47] C.-F. Xu, K. Lewis, K.L. Cantone, P. Khan, C. Donnelly, N. White, N. Crocker, P.R. Boyd, D.V. Zaykin, I.J. Purvis. Effectiveness of computational methods in haplotype prediction. *Human Genetics*, 110: 148-156, 2002.
- [48] J. Xu. Extracting haplotypes from diploid organisms. *Current Issues in Molecular Biology*, 8(2): 113-122, 2006.
- [49] B. V. Halldórsson, V. Bafna, N. Edwards, R. Lippert, S. Yooseph, S. Istrail. Combinatorial problems arising in SNP and haplotype analysis. *DMTCS*: 26-47, 2003.
- [50] D. Gusfield, S.H. Orzack. Haplotype inference. *CRC Handbook of Computational Molecular Biology*. Ed. S.Aluru, CRC Press. 1 ed.: (18-1)-(18-27), 2006.
- [51] D. Gusfield. An overview of combinatorial methods for haplotype inference. Computational methods for SNPs and haplotype inference. *Lecture Notes in Computer Science*: 9-25, 2002.
- [52] R.H. Chung, D. Gusfield. Empirical exploration of perfect phylogeny haplotyping and haplotypers. In *Proc. of the 2003 Cocoon Conference*: 5-19, 2003.

- [53] P. Bonizzoni, G.D. Vedova, R. Dondi, J. Li. The Haplotyping Problem: An Overview of computational models and solutions. *Journal of Computer Science and Technology*. 18(6): 675 – 688, 2003.
- [54] A. Clark. Inference of haplotypes from pcr-amplified samples of diploid populations. *Molecular Biology and Evolution*, 7(2): 111- 122, 1990.
- [55] K. Lange. *Mathematical and statistical methods for genetic analysis*. 2nd ed. Springer-Verlag, New York, 2002.
- [56] L.,Excoffier, M. Slatkin. Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology and Evolution*, 12(5): 921-927, 1995.
- [57] M. Stephens, N.J. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *American Journal of Human Genetics*, 68: 978-989, 2001.
- [58] T. Niu, Z.S. Qin, X. Xu, and J.S. Liu. Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *American Journal of Human Genetics*, 710: 157-169, 2002.
- [59] G. Kimmel, R. Shamir. The incomplete perfect phylogeny haplotype problem. *Journal of Bioinformatics and Computational Biology* 3(2): 359-384, 2005.
- [60] D. Gusfield. Inference of haplotypes from samples of diploid populations: complexity and algorithms. *Journal of Computational Biology*, 8(3): 305 -323, 2001.
- [61] D. Gusfield. Haplotype inference by pure parsimony. *Lecture Notes in Computer Science*, 2676: 144-155, 2003.
- [62] D. Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. In *Proc. of 6th Annual Conference on Research in Computational Molecular Biology (RECOMB 2002)*: 166-175, 2002.
- [63] E. Halperin, E. Eskin, and R. M. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. *Journal of Bioinformatics and Computational Biology*, 1 (1) 1-20, 2003.
- [64] V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: A direct approach. *Journal of Computational Biology*, 10(3-4):323-40, 2003.
- [65] L. Wang, Y. Xu. Haplotype inference by maximum parsimony. *Bioinformatics*, 19(14): 1773-1780, 2003.
- [66] R.-S. Wang, X.-S. Zhang. L. Sheng. Haplotype inference by pure parsimony via genetic algorithm. *Operations Research and Its Applications in Lecture Notes in Operations Research*, 5: 308-318, 2005.
- [67] G. Lancia, C. Pinotti, R. Rizzi. Haplotyping populations by pure parsimony: complexity, exact and approximation algorithms. *INFORMS Journal of Computing*, 16(4): 348-359, 2004.
- [68] Y.-T. Huang, K.-M. Chao, T. Chen. An approximation algorithm for haplotype inference by maximum parsimony. *Journal of Computational Biology*, 12(10) : 1261 -1274, 2005.
- [69] The International HapMap Consortium. The international HapMap project. *Nature* 426: 789-796, 2003.
- [70] The International HapMap Consortium. A haplotype map of the human genome. *Nature* 437: 1299-1320, 2005.
- [71] D. Qian, L. Beckmann. Minimum-recombinant haplotyping in pedigrees. *American Journal of Human Genetics*, 70(6): 1434-1445, 2002.

- [72] S. Lin, T.P Speed. An algorithm for haplotype analysis. *Journal of Computational Biology*, 4(4): 535–546, 1997.
- [73] J. Li, T. Jiang. Efficient rule-based haplotyping algorithms for pedigree data. In *Proc. of 7th Annual Conference on Research in Computational Molecular Biology (RECOMB 2003)*: 197-206, 2003.
- [74] J. Li, T. Jiang. Efficient inference of haplotypes from genotypes on a pedigree. *Journal of Bioinformatics and Computational Biology*, 1(1): 41-69, 2003.
- [75] K. Doi, J. Li, T. Jiang. Minimum recombinant haplotype configuration on tree pedigrees. *Lecture Notes in Computer Science*, 2812: 339-353, 2003.
- [76] J. Li, T. Jiang. *PedPhase: Haplotype Inference for Pedigree Data*. 2003. Accessed on 14 May 2007 at <<http://vorlon.case.edu/~jx1175/PedPhase.pdf>>
- [77] P. Tapadar, S. Ghosh, P.P. Majumder. Haplotyping in pedigrees via a genetic algorithm. *Human Heredity*, 50: 43-56, 2000.
- [78] K. Zhang, J. Li. HaploBlockFinder: haplotype block analyses. *Bioinformatics*, 19(10): 1300–1301, 2003.
- [79] B. Devlin, N. Risch. A comparison of linkage disequilibrium measures for fine-scale mapping. *Genomics*, 29: 311-322, 1995.
- [80] *Approaches to gene mapping in complex human diseases*. Ed. Haines, J.L., Pericak-Vance, M.A. Wiley-Liss, Inc., New York. 1st ed. 1998.
- [81] P. Sheet, M. Stephens. A Fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. *The American Journal of Human Genetics*, 78: 629-644, 2006.
- [82] S. Sun, C.M.T. Greenwood, R.M. Neal. Haplotype inference using a Bayesian Hidden Markov model. *Genetic Epidemiology*, accepted 13 July 2007.
- [83] E. Halperin, E. Eskin, R.M. Karp. Large scale reconstruction of haplotypes from genotype data. In *Proc. of 7th Annual Conference on Research in Computational Molecular Biology (RECOMB 2003)*: 104-113, 2003.
- [84] G. Kimmel, R. Shamir. Maximum likelihood resolution of multi-block genotypes. In *Proc. of the eighth annual international conference on Research in computational molecular biology*, San Diego, California, USA.: 2-9, 2004.
- [85] G. Greenspan, D. Geiger. Model-based inference of haplotype block variation. In *Proc. of The Seventh Annual International Conference on Research in Computational Molecular Biology (RECOMB 2003)*: 131–137, 2003.
- [86] E. Halperin, E. Eskin. Haplotype reconstruction from genotype data using Imperfect Phylogeny. *Bioinformatics*, 20(12): 1842-1849, 2004.
- [87] T.D. Petes. Meiotic recombination hot spots and cold spots. *Nature Reviews Genetics*, 2: 360-369, 2001.
- [88] K. Zhang, Z.S. Qin, J.S. Liu, T. Chen, W.S. Waterman, F. Sun. Haplotype block partitioning and tag SNP selection using genotype data and their applications to association studies. *Genome Research* 14(5),908-16, 2004.
- [89] B. J. Hayes, P. M. Visscher, H. C. McPartlan, M. E. Goddard. Novel multilocus measure of linkage disequilibrium to estimate past effective population size. *Genome Research*, 13(4): 635-643, 2003.

- [90] C. Sabatti, N. Risch. Homozygosity and linkage disequilibrium. *Genetics*, 160(4): 1707-1719, 2002.
- [91] R. Gorelick, M.D. Laubichler. Decomposing multilocus linkage disequilibrium. *Genetics*, 166: 1581-1583, 2004.
- [92] W.J. Ewens, G.R. Grant. *Statistical Methods in Bioinformatics: An Introduction*. Springer; 2nd ed., 2005.
- [93] Eberhart, R.C., Dobbins, R., Simpson, P. *Computational intelligence PC tools*. AP Professional, 1996.
- [94] R.L. Haupt, S.E. Haupt. *Practical Genetic Algorithms*. Wiley-Interscience, 2004.
- [95] M. Safe, J. Carballido, I. Ponzoni, N. Brignole. On stopping criteria for genetic algorithms. *Lecture Notes in Computer Science*, 3171: 405-413, 2004.
- [96] D. Greenhalgh, S. Marshall. Convergence criteria for genetic algorithms. *SICOMP*, 30(1): 269-282, 2000.
- [97] Drysdale, C., McGraw, D., Stack, C., Stephens, J., Judson, R., Nandabalan, K., Arnold, K., Ruano, G. and Liggett, S. Complex promoter and coding region β 2-adrenergic receptor haplotypes alter receptor expression and predict in vivo responsiveness. In *Proc. of the National Academy of Sciences USA*, 97: 10483–10488, 2000.
- [98] Rieder, M., Taylor, S., Clark, A. and Nickerson, D. Sequence variation in the human angiotensin converting enzyme. *Nature Genetics*, 22: 59–62, 1999.
- [99] *The HapMap project*. Accessed on 12 August 2007 at <<http://www.hapmap.org>>
- [100] M. Stephens, P. Donnelly. A comparison of Bayesian methods for haplotype reconstruction from population genotype data. *American Journal of Human Genetics*, 73: 1162-1169, 2003.
- [101] J.C. Barrett, B. Fry, J. Maller, M.J. Daly. Haploview: analysis and visualization of LD and haplotype maps. *Bioinformatics*, 21: 263–265, 2005.
- [102] Developing a haplotype map of the human genome for finding genes related to health and disease. *Haplotype Map Development Report*, Washington, D.C. July 18-19, 2001. Accessed on 17 May 2007 at <<http://www.genome.gov/10001665>>
- [103] D.B. Goldstein, M.E. Weale. Population genomics: linkage disequilibrium holds the key. *Current Biology*, 11: R576-579, 2001.
- [104] N. Liu, S. Sawyer, N. Mukherjee, A. Pakstis, J. Kidd, K. Kidd, A. Brookes, H. Zhao. Haplotype block structures show significant variation among populations. *Genetic Epidemiology*, 2004, 27: 385–400.
- [105] S. Gu, A. Pakstis, H. Li, W. Speed, J. Kidd, K. Kidd. Significant variation in haplotype block structure but conservation in tagSNP patterns among global populations. *European Journal of Human Genetics*, 15(3): 302–312, 2007.
- [106] I. Menashe, O. Man, D. Lancet, Y. Gilad. Population differences in haplotype structure within a human olfactory receptor gene cluster. *Human Molecular Genetics*, 11(12): 1381–1390, 2002.
- [107] E. P. Xing, K.-A. Sohn, M. I. Jordan, Y. W. Teh: Bayesian multi-population haplotype inference via a hierarchical dirichlet process mixture. In *Proc. of the 23rd international conference on Machine learning*: 1049-1056, 2006.
- [108] E. Xing, R. Sharan, M. Jordan. Bayesian haplotype inference via the Dirichlet process. In *Proc. of the 21st International Conference on Machine Learning*: 99-112, 2004.

Appendix A

R functions for the results evaluation

```
ham.dist <- function(s1, s2) { # Function to compute the hamming distance
of the two strings of the same length
# Positions with values 2 or 9 in the first (true) string are NOT COUNTED
dist <- 0
  for (j in 1:length(s1))
  {
    if (s1[j]!=9 && s1[j]!=2)
    {
      if (s1[j]!=s2[j]) dist <- dist+1
    }
  }
return(dist)
}
```

```
block.err.star <- function(Gen.coded.df, true.haplo.df, resolved.df, b){
# Gencoded.df contains input genotypes coded into 0,1,2 and 9, nrow = n
# true.haplo.df has the true resolutions (where possible) of the input
genotypes, nrow = 2n
# resolved.df has computed resolution of the input genotypes, nrow = 2n
# b is the vector indicating the block structure (at the beginning
position of each block is the value of the ending position)
# Missing data is ignored for the purpose of calculating the error rates
tote <- 0
totr <- 0
err <- c() # Error rates for each block (associated with the beginning
positions of blocks)
# Initializing vector err:
for (j in 1:length(b)) err[j] <- 0
e <-c() # Number of errors for each genotype within a block
r <- c() # Number of heterozygotes (resolvable) in each genotypes within a
block
  g <- c() # Current genotype
  h1 <- c() # Predicted resolutions h1 & h2
  h2 <- c()
  t1 <- c() # True resolutions t1 & t2
  t2 <- c()
begin <- 1
while (begin < (length(b)+1))
{
  end <- b[begin]
  for (i in 1:nrow(Gen.coded.df)) # For each genotype in the
input:
  {
    g <- Gen.coded.df[i, begin:end]
    t1 <- true.haplo.df[(2*i-1), begin:end]
    t2 <- true.haplo.df[2*i, begin:end]
    h1 <- resolved.df[(2*i-1), begin:end]
    h2 <- resolved.df[2*i, begin:end]
    # Counting the number of resolvable heterogeneous positions
    #r[i] <- as.numeric(sum(g==2)) - as.numeric(sum(t1==2))
    #Counting the number of ALL heterogeneous positions
    r[i] <- as.numeric(sum(g==2))
  }
}
```

```

        # Computing the number of errors (using Hamming distance):
        e[i] <- 0.5*min((ham.dist(t1,h1)+ham.dist(t2,h2)),
(ham.dist(t1,h2)+ham.dist(t2,h1)))
    }
    #print(sum(r))
    err[begin] <- sum(e)/sum(r)
    tote <- tote+sum(e)
    totr <- totr+sum(r)
    begin <- end+1
  }
print('Average block error rate: ')
ave.rate <- tote/totr
print(ave.rate)
return(err)
}

switch.rate <- function(Gen.coded.df, true.haplo.df, resolved.df){
switches <- c()
heter <- 0
for (i in 1:nrow(Gen.coded.df))
{
  switches[i] <- 0
  k <- 1
  find <- 0
  while (find==0 && k<ncol(Gen.coded.df))
  {
    if (Gen.coded.df[i,k]==2)
    {
      heter <- heter+1
      if (true.haplo.df[2*i,k]!=2) find <- 1
      #Start calculating switches
    }
    k <- k+1
  }
  if (true.haplo.df[2*i,(k-1)]==resolved.df[2*i,(k-1)])
  {t <- 2*i
   h <- 2*i }
  else {t <- 2*i
        h <- 2*i-1 }
  for (j in k:ncol(Gen.coded.df))
  {
    if (Gen.coded.df[i,j]==2)
    {
      heter <- heter +1
      if (true.haplo.df[t,j]!=2 &&
true.haplo.df[t,j]!=resolved.df[h,j])
      {
        switches[i] <- switches[i]+1
        if (h == 2*i) h<- 2*i-1
        else h <- 2*i
      }
    }
  }
}
rate <- sum(switches)/heter
return(list(rate=rate, switches=switches))
}

```

Appendix B

R code for the Daly data preprocessing

```
# Download data
prime.data <- scan(file="C:/Documents and Settings/Owner/My
Documents/dissertation/data/daly/prime_data.dat", what="", sep="\n")

#Need to exclude the empty spaces and tabulations as well:
new.data <- prime.data
n <- length(prime.data) # This is simply the number of rows read
for (i in 1:n) {
  t <- 2
  k <- nchar(prime.data[i])
  while (t<(k+1)) {
    if ((substring(new.data[i],t,t) == "\t") || (substring(new.data[i],t,t)
    == " ")) { # Change quotes in R !
      new.data[i] <- paste(substring(new.data[i],1,(t-
      1)),substring(new.data[i],(t+1),k), sep="") # Change quotes in R !
    }
    else t <- t+1
  }
  k <- nchar(new.data[i])
}

#Separate the string into a vector with nchar/2 elements
for (j in 1:n) {
  v <- c()
  k <- nchar(new.data[j])/2
  i <- 1
  while (i < (k+1)) {
    v[i] <- substring(new.data[j],(2*i-1),2*i)
    i <- i+1
  }
  if (j==1) new <- v
  else new <- rbind(new, v)
}
x <- c(1:387)
row.names(new) <- x
new <- as.data.frame(new)

#Add ID's to the data frame:
ID <- scan(file="C:/Documents and Settings/Owner/My
Documents/dissertation/data/daly/ID.dat", what="", sep="\n")
ID.father <- scan(file="C:/Documents and Settings/Owner/My
Documents/dissertation/data/daly/ID_father.dat", what="", sep="\n")
ID.mother <- scan(file="C:/Documents and Settings/Owner/My
Documents/dissertation/data/daly/ID_mother.dat", what="", sep="\n")
family <- scan(file="C:/Documents and Settings/Owner/My
Documents/dissertation/data/daly/family.dat", what="", sep="\n")

new.df <- cbind (family, ID, ID.father, ID.mother, new)

#Mark all children:
child <- c()
for (i in 1:387){
```

```

if (new.df$ID.father[i] != 0) { # Change quotes in R
child[i] <- "child"}
else child[i] <- "parent" # Change quotes in R
}
new.df <- cbind(child, new.df)

#Preprocessing itself. The working data frame is new.df
#Rule: in the decoded sequence first position is father's, second is
mother's.
#Arranging all genotypes in the increasing order for easier handling:
geno.df <- new.df
for (i in 1:387){
for (j in 6:108){
v <-
c(as.numeric(substring(geno.df[i,j],1,1)),as.numeric(substring(geno.df[i,j]
],2,2)))
v <- sort(v)
geno.df[i,j] <- paste(v[1],v[2],sep="")
}
}
haplo.df <- data.frame()
for (i in 1:387){ # begin for
if (geno.df$child[i] == "child"){ # begin if
child <- geno.df[geno.df$ID == geno.df$ID[i], ]
father <- geno.df[geno.df$ID == geno.df$ID.father[i], ]
mother <- geno.df[geno.df$ID == geno.df$ID.mother[i], ]

for (j in 6:108){ # begin for
# for (j in 1:12){ # This is just for the test data
if (substring(child[[j]],1,1) != substring(child[[j]],2,2)) {
# For ambiguous sites only
if (father[[j]]!="00" && mother[[j]]!="00") { # At least one of the
parents must be non-missing
#Assuming there is no missing information
if (father[[j]]==mother[[j]]) child[j] <- paste(" ",child[[j]],sep="")
else
{
if (substring(father[[j]],1,1)!=substring(father[[j]],2,2) ||
substring(mother[[j]],1,1)!=substring(mother[[j]],2,2))
{
# Case 1:
if (child[[j]]==father[[j]] &&
as.numeric(substring(mother[[j]],1,1))>as.numeric(substring(father[[j]],1,
1)))
{ child[j] <- father[[j]] }
else if (child[[j]]==father[[j]]) child[j] <-
paste(substring(father[[j]],2,2), substring(mother[[j]],1,1),sep="")
else if
(as.numeric(substring(father[[j]],1,1))>as.numeric(substring(mother[[j]],1
,1))) # child = mother
{ child[j] <-
paste(substring(father[[j]],2,2), substring(mother[[j]],1,1),sep="") }
else child[j] <- mother[[j]]
}
}
# Case 2
else if (substring(father[[j]],1,1)==substring(father[[j]],2,2))

```

```

        { child[j] <- paste(substring(father[[j]],1,1),
substring(mother[[j]],2,2),sep="") }
        else child[j] <- paste("*",child[[j]],sep="")
    }

}
# Case 3: mother or father has missing info
else if (father[[j]]!=mother[[j]]) # either father or mother has non-
missing info
    { if (father[[j]]!="00" && father[[j]]!=child[[j]])
        # I.e. father has non-missing unambiguous info, like "44, and child
is ambiguous"
        { if
(substring(father[[j]],1,1)!=substring(child[[j]],1,1))
            {child[j] <- paste(substring(child[[j]],2,2),
substring(child[[j]],1,1),sep="") } # else nothing changes
            }
        if (mother[[j]]!="00" && mother[[j]]!=child[[j]])
            # I.e. mother has non-missing unambiguous info, like
"44, and child is ambiguous"
            { if
(substring(mother[[j]],1,1)==substring(child[[j]],1,1))
                {child[j] <- paste(substring(child[[j]],2,2),
substring(child[[j]],1,1),sep="") } # else nothing changes
                }
            if (mother[[j]]==child[[j]] || father[[j]]==child[[j]]) #
non-missing but ambiguous info in one parent, other missing
                { child[j] <- paste("*",child[[j]],sep="") }
            }
        else child[j] <- paste("*",child[[j]],sep="")
    }
} # end for
haplo.df <- rbind(haplo.df,child)
} # end if
} # end for

#Count unresolved (ambiguous) sites and missing entries in rows:
missing <- c()
ambig <- c()
total <- c()
for (i in 1:nrow(haplo.df)) {
    mis <- 0
    amb <- 0
    tot <- 0
    for (j in 6:108) {
        if (haplo.df[i,j]== "00") mis <- mis +1
        if (substring(haplo.df[i,j],1,1)=="*") amb <- amb+1
    }
    missing[i] <- mis
    ambig[i] <- amb
    total[i] <- mis+amb
}
haplo.df <- cbind(haplo.df, missing, ambig, total)
# percentage missing and ambiguous
mis.per <- missing/103
amb.per <- ambig/103
tot.per <- total/103

```

```

#Now, the data needs to be properly encoded (as 0,1, 2 or 9) with
appropriate vector storing coded information

#First, remove "*" everywhere and store the data in the new df:
#haplo.df contains markers ("*") of unresolvable genotype positions and is
needed for the verification purposes

Gen.orig.df <- haplo.df
for (j in 6:108) {
  for (i in 1:129) { # With 2 individuals (86 and 105) put back
    if (substring(Gen.orig.df[i,j],1,1)=="*") Gen.orig.df[i,j] <-
      substring(Gen.orig.df[i,j],2,3)
  }
}

#Then create vector (2 positions each element) storing coded information:
"0" (listed first) and "1" (listed second):
#(It just happened so that the 3rd position of the levels() always has 2
different values):

codes <- c()
for (j in 6:108) { # Searching for different symbols
  s1<-substring(levels(as.factor(Gen.orig.df[,j]))[3],1,1)# coded by 0
  s2<-substring(levels(as.factor(Gen.orig.df[,j]))[3],2,2)# coded by 1
  nextcode <- paste(s1,s2,sep="")
  codes <- c(codes, nextcode)
}

#Code genotype into the 0,1,2 and 9:

Gen.coded.df <- c()
for (i in 1:129){
  newrow <- c()
  for (j in 6:108){
    if
      (substring(Gen.orig.df[i,j],1,1)==substring(Gen.orig.df[i,j],2,2))
    {
      if (substring(Gen.orig.df[i,j],1,1=="0") newrow[j-5]
<- 9 # Missing data
      else
      {
        if
          (substring(Gen.orig.df[i,j],1,1)==substring(codes[j-5],1,1)) newrow[j-5]
<- 0
        else newrow[j-5] <- 1
      }
    }
    else # heterogeneous positions
    { newrow[j-5] <- 2}
  }
  Gen.coded.df <- rbind(Gen.coded.df, newrow)
}
rownames(Gen.coded.df) <- c(1:129)

#Create the true haplotype matrix (it also contains unresolvable sites)
true.haplo.df <- c()

```

```

# New data frame containing resolved haplotypes (every 2 haplotypes/rows
correspond to 1 genotype)
for (i in 1:129){
h1 <- c()
h2 <- c()
for (j in 6:108){
  if (substring(haplo.df[i,j],1,1)=="*") # Unresolvable position
  {
    h1[j-5] <- 2
    h2[j-5] <- 2
  }
  else
  {
    if
(substring(haplo.df[i,j],1,1)==substring(haplo.df[i,j],2,2)) # homogeneous
positions
    {
      if (substring(haplo.df[i,j],1,1)=="0")
      {
        h1[j-5] <- 9 # Missing unresolvable data
        h2[j-5] <- 9
      }
      else
      {
        if
(substring(haplo.df[i,j],1,1)==substring(codes[j-5],1,1))
        {
          h1[j-5] <- 0
          h2[j-5] <- 0
        }
        else
        {
          h1[j-5] <- 1
          h2[j-5] <- 1
        }
      }
    }
  }
  else # heterogeneous positions
  {
    if (substring(haplo.df[i,j],1,1)==substring(codes[j-5],1,1))
    {
      h1[j-5] <- 0
      h2[j-5] <- 1
    }
    else
    {
      h1[j-5] <- 1
      h2[j-5] <- 0
    }
  }
}
}
true.haplo.df <- rbind(true.haplo.df, h1, h2)
}

```

Appendix C

R code for the Patil data preprocessing

#Copying b blocks:

```
b <- 1000 # Number of blocks to be read
Patil <- scan(file="C:/Documents and Settings/Owner/My
Documents/dissertation/data/Perlegen/Patil_Ch21/haploptype.txt", what="",
sep="\n", skip=1, nlines=44*b)

Patil.df <- c()
for (i in 1:b)
{
  Patil.data <- Patil[(22*(i-1)+1):(22*i)]
  n <- length(Patil.data) # This is simply the number of rows read
  (22 or 20)
  P.df <- rep('*',n)
  for (i in 1:n) {
    numtab <- 0 # Number of tabulation symbols
    t <- 2
    k <- nchar(Patil.data[i])
    stop = 0
    while (t<(k+1) && stop==0)
    {
      if (substring(Patil.data[i],t,t) == '\t') numtab <- numtab+1
      if (numtab==3)
      {
        P.df[i] <- paste(substring(P.df[i],1,(t-
1)),substring(Patil.data[i],(t+1),(t+1)), sep='')
      }
      if (numtab==4) stop=1
      t <- t+1
    }
    P.df[i] <- substring(P.df[i],2,(nchar(P.df[i])-1))
  }
  Patil.df <- cbind(Patil.df,P.df)
}
```

#Then, transform data into the single strings:

```
patil.df <- c()
for (i in 1:nrow(Patil.df))
{
  patil.df[i] <- Patil.df[i,1]
  for (j in 1: ncol(Patil.df))
  {
    patil.df[i] <- paste(patil.df[i],Patil.df[i,j],sep='')
  }
}
```

#Separating the string into a vector with nchar elements:

```
patil <- c()
for (j in 1:n) {
```

```

k <- nchar(patil.df[j])
v <- c()
i <- 1
while (i < (k+1))
{
    v[i] <- substring(patil.df[j],i,i)
    i <- i+1
}
patil <- rbind(patil, v)
}
x <- c(1:length(patil.df))
row.names(patil) <- x

patil <- as.data.frame(patil)

#Encoding into 0,1,9:

patil.codes <- c()
v <- c()
for (j in 1:ncol(patil))
{
    v <- levels(patil[,j])
    patil.codes <- cbind(patil.codes, v)
}

for (i in 1:2) # Rearranging
{
    for (j in 1:ncol(patil))
    {
        if (patil.codes[i,j]=='n')
        {
            v <- patil.codes[3,j]
            patil.codes[3,j] <- 'n'
            patil.codes[i,j] <- v
        }
    }
}

# Encoding: first row in patil.codes is codes into 1's, second row into 0's, 'n' into 9's
patil.coded <- matrix(nrow=nrow(patil), ncol=ncol(patil))
for (i in 1:nrow(patil))
{
    for (j in 1:ncol(patil))
    {
        if (patil[i,j]=='n') patil.coded[i,j] <- 9
        else if (patil[i,j]==patil.codes[1,j]) patil.coded[i,j] <- 1
        else patil.coded[i,j] <- 0
    }
}

#Before inputting the data into the algorithm it has to be simulated
 #(random matching of the pairs of haplotypes to create #genotypes):

# Delete unwanted rows
# Rows 6 and 22 also don't have enough defined data: so they all should be
deleted:

```

```

set <- c(6,22)
patil.coded <- patil.coded[-set,]

# True Haplotype matrix
v <- 1:20
patil.haplo <- c()
for (i in 1:100) # Generate 100 individuals/genotypes
{
  set <- sample(v,2)
  patil.haplo <- rbind(patil.haplo, patil.coded[set[1],],
    patil.coded[set[2],])
}

patil.geno <- c()
g <- c()
for (i in 1:100)
{
  for (j in 1:ncol(patil.coded))
  {
    if (patil.haplo[(2*i-1),j]==patil.haplo[2*i,j]
    &&patil.haplo[2*i,j]!=9) g[j] <- patil.haplo[2*i,j]
    else if (patil.haplo[(2*i-1),j]==9 || patil.haplo[2*i,j]==9)
    g[j] <- 9
    else g[j] <- 2
  }
  patil.geno <- rbind(patil.geno, g)
}

```

Appendix D

Documentation for *haplogen* package

D.1 Introduction

Haplogen.pack is an R package implementing two algorithms for haplotyping and block partitioning: HAPLOGEN and HAPLOCLUST.

Algorithm HAPLOGEN is designed to find phased haplotypes and the associated haplotype block partition for the given collection of genotypes. HAPLOGEN find the full haplotype block profile consisting of the collections of haplotype patterns and their sample frequencies for every block. In addition, the algorithm determines the scores for block boundaries which estimate how likely for every position to have a block boundary immediately to the left of it.

Algorithm HAPLOCLUST is an extension of the algorithm HAPLOGEN. It performs the haplotype resolution and block partition for the mixed population genotype samples (two population case only). Algorithm HAPLOCLUST clusters the given genotype sample into two clusters (if possible) with significantly different block structures and then find haplotype resolution and block structure for both of them. Part of the sample may be left unclustered.

D.2 Getting started

D.2.1 Installation for Windows

Haplogen.pack can be installed like any other R package. Download the compiled binary version in *haplogen.pack_1.0.zip* file containing the package to you computer. The best way is to save it in the **bin** directory of R. Start R, in the menu select **Packages** and then **Install package(s) from local zip files...** option on the bottom.

In the dialog window navigate to the location of the zip file containing the package (i.e., the **bin** directory, as suggested). Click the **Open** button. After that the message about successful installation should appear in the R console.

D.2.2 Usage

After the package was successfully installed you can start using it. In the R console type

```
library(haplogen.pack)
```

and then assuming you have data G in an appropriate format already downloaded into R type:

```
haplogen(G)
haplogen(G, numit=1000) # Specifying the number of iterations
```

if you want to use HAPLOGEN algorithm

or

`haplogen(G, method="CLUST")` if you want to use HAPLOCLUST algorithm.

You can also type `help(haplogen)` to look at the R documentation file about the package which contains brief information on how to use the package as well as some examples.

D.2.3 Removal

If you would like to remove the current version of the `haplogen.pack` package from the library issue the following command in R:

```
remove.packages("haplogen.pack")
```

D.3 Input file format

Sample of genotypes should be represented in R by either a matrix or a data frame. Each genotype should be encoded in a single row by 0, 1 (homogeneous alleles), 2 (ambiguous allele) and 9 (missing data).

The genotype data can easily be downloaded to R by using the following command:

```
G <- read.table("<path to the file>")
```

D.4 Output

Output differs for HAPLOGEN and HAPLOCLUST (when using `method = "CLUST"`) algorithms. All of the output files are stored in current version of R directory.

D.4.1 Output files for HAPLOGEN algorithm

For HAPLOGEN the output consists of the following files:

- **H.** This file contains the whole-length haplotype resolution for each genotype. Every two rows correspond to one genotype in the same order as given in the input file.
- **B.** This file contains the description of the haplotype block structure:

- *vector specifying block boundaries*, where the position of each non-zero entry indicates the beginning of a block and the value of that entry its ending position; for example:

```
12 0 0 0 0 0 0 0 0 0 0 0 0 30 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

The above vector shows that there are 2 blocks found spanning positions 1-12 and 13-30.

- *vector of scores for block boundaries*, indicating the number of times the algorithm selected some specific boundaries, for example:

10 0 0 0 0 3 0 0 0 0 0 1 8 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0

The fractional scores can be obtained by dividing every entry of this vector by its first element (10 in this example).

- *the full haplotype blocks profile* including the list of distinct haplotype patterns within any given block and their sample frequencies (last column). For example:

```

Block 1
Positions 1 through 12

1 1 1 1 1 1 1 1 1 1 1 1      2
1 1 0 1 1 1 1 1 1 1 1 1      4
0 1 1 1 1 1 1 1 1 1 1 1     16
0 1 1 1 1 1 1 0 1 1 1 1      2
1 1 0 1 1 1 0 1 1 1 1 1      4
1 0 1 0 0 0 0 1 1 1 1 1      2
1 1 0 1 1 1 1 0 0 0 0 0      2
1 1 1 0 0 0 0 1 1 1 1 1      2
1 1 1 0 0 0 0 0 0 0 0 0      2

Block 2
Positions 13 through 30

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1      4
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1     12
1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1      2
1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1     10
1 1 0 0 0 1 1 0 0 1 0 1 0 0 1 1      2
0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 1      2
1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1      2
0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1      2

```

D.4.2 Output files for HAPLOCLUST algorithm

For HAPLOCLUST the output consists of the following files:

- **G.** This file contains the original genotype matrix with the additional column (the first column) indicating the cluster assignment: 1 (cluster 1), 2 (cluster 2) or 0 (unclustered genotype).
- **B1, B2.** These files contain the haplotype block profiles for each extracted cluster similar to those produced by HAPLOGEN algorithm.
- **H1, H2.** Files with the haplotype resolutions for the genotypes with corresponding assignment. Every two lines correspond to an individual genotype with the same order as in the input genotype file. Only resolution for genotypes with corresponding assignment (1 or 2) is shown. The rest of the genotypes appear by rows of “5 5 5 5 5 5”.