Department of Computer Science
Series of Publications A
Report A-2009-9

Computational Techniques for Haplotype Inference and for Local Alignment Significance

Pasi Rastas

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Festival Hall, University Language Centre (Fabianinkatu 26), on November 20th, 2009, at 12 o'clock noon.

University of Helsinki
Finland

Contact information

Postal address:
Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
Finland

Email address: postmaster@cs.helsinki.fi (Internet)

URL: http://www.cs.Helsinki.FI/

Telephone: +358 9 1911

Telefax: +358 9 191 51120

ISSN 1238-8645
ISBN 978-952-10-5879-0 (paperback)
ISBN 978-952-10-5880-6 (PDF)
Computing Reviews (1998) Classification: G.3, J.3, G.2.1, F.2.2
Helsinki 2009
Helsinki University Print

Computational Techniques for Haplotype Inference and for Local Alignment Significance

Pasi Rastas

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
Pasi.Rastas@cs.Helsinki.FI
http://www.cs.Helsinki.FI/u/prastas/

PhD Thesis, Series of Publications A, Report A-2009-9
Helsinki, November 2009, xii+64+50 pages
ISSN 1238-8645
ISBN 978-952-10-5879-0 (paperback)
ISBN 978-952-10-5880-6 (PDF)

Abstract

This thesis which consists of an introduction and four peer–reviewed original publications studies the problems of haplotype inference (haplotyping) and local alignment significance. The problems studied here belong to the broad area of bioinformatics and computational biology. The presented solutions are computationally fast and accurate, which makes them practical in high–throughput sequence data analysis.

Haplotype inference is a computational problem where the goal is to estimate haplotypes from a sample of genotypes as accurately as possible. This problem is important as the direct measurement of haplotypes is difficult, whereas the genotypes are easier to quantify. Haplotypes are the key–players when studying for example the genetic causes of diseases. In this thesis, three methods are presented for the haplotype inference problem referred to as HaploParser, HIT, and BACH.

HaploParser is based on a combinatorial mosaic model and hierarchical parsing that together mimic recombinations and point–mutations in a biologically plausible way. In this mosaic model, the current population is assumed to be evolved from a small founder population. Thus, the haplotypes of the current population are recombinations of the (implicit) founder haplotypes with some point–mutations.

HIT (Haplotype Inference Technique) uses a hidden Markov model for haplotypes and efficient algorithms are presented to learn this model from genotype data. The model structure of HIT is analogous to the mosaic model of HaploParser with founder haplotypes. Therefore, it can be seen as a probabilistic model of recombinations and point–mutations.

BACH (Bayesian Context–based Haplotyping) utilizes a context tree weighting algorithm to efficiently sum over all variable–length Markov chains to evaluate the posterior probability of a haplotype configuration. Algorithms are presented that find haplotype configurations with high posterior probability. BACH is the most accurate method presented in this thesis and has comparable performance to the best available software for haplotype inference.

Local alignment significance is a computational problem where one is interested in whether the local similarities in two sequences are due to the fact that the sequences are related or just by chance. Similarity of sequences is measured by their best local alignment score and from that, a p-value is computed. This p-value is the probability of picking two sequences from the null model that have as good or better best local alignment score. Local alignment significance is used routinely for example in homology searches.

In this thesis, a general framework is sketched that allows one to compute a tight upper bound for the p–value of a local pairwise alignment score. Unlike the previous methods, the presented framework is not affeced by so–called edge–effects and can handle gaps (deletions and insertions) without troublesome sampling and curve fitting.

Computing Reviews (1998) Categories and Subject Descriptors:
G.3 [Probability and Statistics]: Markov Processes, Probabilistic Algorithms, Statistical Computing
J.3 [Life and Medical Sciences]: Biology and Genetics
G.2.1 [Discrete Mathematics]: Combinatorics – Combinatorial Algorithms
F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems – Pattern matching

General Terms:
Thesis, Algorithms, Bioinformatics, Computational Biology

Additional Key Words and Phrases:
Haplotype inference, Haplotyping, Genotypes, Phasing, Markov models, Local alignment significance, Significance testing, DNA, SNPs, Hidden Markov models, Markov chains, Variable order Markov chains, Context tree weighting, EM algorithm, Dynamic programming

Acknowledgements

First of all, I am most thankful to my supervisor Esko Ukkonen who has guided me through all these years in the University. His broad scientific experience, patience, and encouragement made this thesis possible.

I am also grateful to Mikko Koivisto, Heikki Mannila, and Jussi Kollin for scientific collaboration included in this thesis. I thank Tapio Salakoski, Tero Aittokallio, and Leena Salmela for their valuable comments and suggestions regarding this thesis.

I would like to thank Heikki Lokki for organizing me a summer job in 2002. I have worked for Esko Ukkonen ever since. I also thank Teemu Kivioja for the quidance and shared knowledge especially during the first years at the Computer Science Department.

Special thanks go to the Computer Science Department and the Helsinki Institute for Information Technology (HIIT). Their computing facilities staff has also been helpful, especially Pekka Niklander and Pekka Tonteri. For the funding I would like to thank the graduate school ComBi, the Finnish Center of Exellence for Algorithmic Data Analysis Research (Algodan), and the European Union. I also thank Marina Kurtén for cheking my English.

The friends at work deserve my sincere thanks. Lunch, coffee, (Bio)beer, and talks with the following people have been valuable: Ari Rantanen, Esa Pitkänen, François Nicolas, Janne Korhonen, Janne Ravantti, Jussi Lindgren, Kimmo Palin (especially for the rope), Markus Heinonen, Matti Kääriäinen, Taneli Mielikäinen, Veli Mäkinen, and the ones I already mentioned or forgot to mention.

I would also like to thank my parents Anitta and Martti, and parents–in–law Aira and Juhani.

Finally, I would like to thank my wife Päivi who has given me unconditional support when it has been difficult.

Helsinki 2.11.2009
Pasi Rastas

Original Publications of the Thesis

This thesis is based on the following peer–reviewed publications, which are referred to as Paper I–IV in the text.

I. Pasi Rastas and Esko Ukkonen:
Haplotype Inference via Hierarchical Genotype Parsing.
In Proc. Algorithms in Bioinformatics, 7th International Workshop, WABI 2007 (Philadelphia, PA, USA), LNBI 4645, pages 85–97.

II. Pasi Rastas, Mikko Koivisto, Heikki Mannila, and Esko Ukkonen:
A Hidden Markov Technique for Haplotype Reconstruction.
In Proc. Algorithms in Bioinformatics, 5th International Workshop, WABI 2005 (Mallorca, Spain), LNBI 3692, pages 140–151.

III. Pasi Rastas, Jussi Kollin, and Mikko Koivisto:
Fast Bayesian Haplotype Inference via Context Tree Weighting.
In Proc. Algorithms in Bioinformatics, 8th International Workshop, WABI 2008 (Karlsruhe, Germany), LNBI 5251, pages 259–270.

IV. Pasi Rastas:
A General Framework for Local Pairwise Alignment Statistics with Gaps.
In Proc. Algorithms in Bioinformatics, 9th International Workshop, WABI 2009 (Philadelphia, PA, USA), LNBI 5724, pages 233–245.

Contents

Chapter 1

Introduction

We begin with the basics of genetics and the background of this work. Then the problems of haplotype inference and local alignment significance are introduced in more detail, followed by a listing of the main contributions included in this thesis.

1.1 Genetics and Genetic Sequences

The hereditary genetic information of an organism is stored in the *genome* [52]. The genome is typically encoded in *DNA* (deoxyribonucleic acid), a nucleic acid that can be described as a string of *nucleotide bases*, adenine (A), cytosine (C), guanine (G), and thymine (T). The actual DNA molecule consists of *complementary strands* where A is always paired with T and C is always paired with G and vice–versa. In a few viruses, the genome is encoded as a nucleic acid *RNA* (ribonucleic acid). RNA differs from DNA by containing uracil (U) instead of thymine (T).

The genome of an organism is structured into chromosomes containing *genes*; parts that affect the function of a cell. Most genes encode *proteins* that act as enzymes, receptors, storage proteins, transport proteins, transcription factors, signaling molecules, hormones, etc.

A position within a chromosome is called a *locus* or a *site*. A locus that contains variation among individuals is called a *marker*. A common type of marker is a single nucleotide polymorphism (SNP). An SNP is a single locus in which there exist at least two possible nucleotides or *alleles* among the population. Other types of markers are for example the number of repeats at *microsatellite* markers.

In the cells of a diploid organism, e.g. a human being, chromosomes are structured into similar chromosome pairs, *autosomes*. The pairs or copies

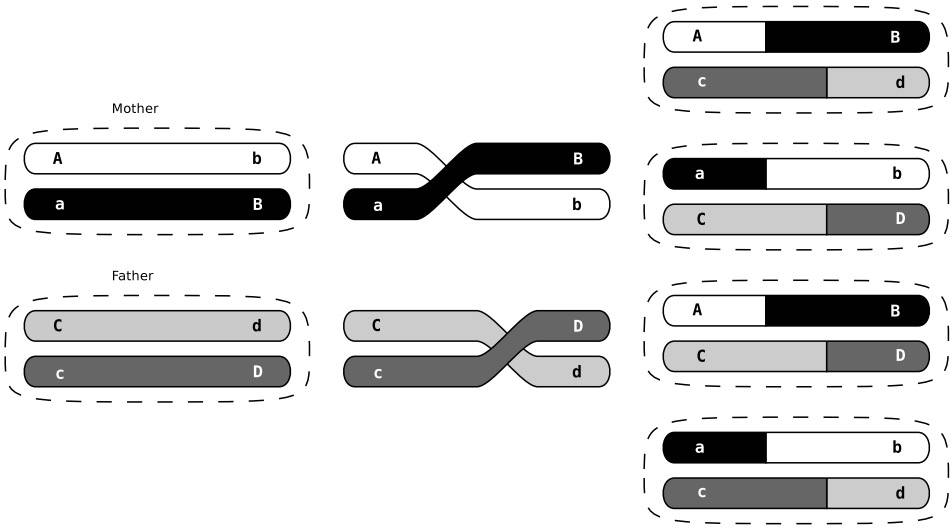

Figure 1.1: Four possibilities for the child's chromosomes (haplotypes) on right when the parents' chromosomes (left) undergo two recombination events in meiosis.

of the autosome are inherited from the individual's parents. One copy is inherited from the mother and the other copy is inherited from the father. Chromosomes of a chromosome pair are, except for the sex chromosomes, very similar among pairs and individuals.

Haploid germline cells, e.g. sperm and egg cells, have only one copy of each chromosome. These cells are formed in *meiosis*, where the genetic material will be selected (to be inherited by a child) from the parents' two copies of each chromosome pair. In a meiosis, a major factor combining the chromosomal information is the *crossing over*, a mechanism to create (genetic) *recombination*. The recombination combines parents' chromosome pairs so that a child may inherit parts of both copies of each chromosome. The rate of recombination in humans is on average about one recombination per generation (meiosis) per 80 million ($8 \cdot 10^7$) base pairs [64, 26] (1.26 cM/Mb, cM=centimorgan). Recombination is illustrated in Figure 1.1.

The genetic material may be altered also in *mutations*. Different types of mutations include *point mutations* (one nucleotide changes), *deletion* and *insertion* of a single or many nucleotides. The typical rate of mutation in humans is 10^{-9} to 10^{-11} per nucleotide per cell division [64].

Markers:	1	2	3	
DNA:	...atatgtcc**C**tat...	ctgt**C**aag...	ctgg**A**ctacgcgt...	(maternal)
	...atatgtcc**G**tat...	ctgt**C**aag...	ctgg**T**ctacgcgt...	(paternal)
Haplotypes:	**G**	**C**	**A**	= 1 0 0
	C	**C**	**T**	= 0 0 1
Genotype:	**C,G**	**C,C**	**A,T**	= 2 0 2

Figure 1.2: An example of SNP haplotypes and an SNP genotype of a single individual over three markers. On the right hand side, the haplotypes are described as binary strings and the genotype is described as a ternary string of 0,1, and 2. The idea for this figure is from [37].

1.1.1 Haplotypes and Genotypes

A *haplotype* consists of an individual's alleles inherited from one parent, thus each individual has two haplotypes. A *genotype* contains information of both haplotypes without the information of which alleles are inherited together from the same parent. The genotypes are easy to measure, whereas haplotypes are laborious, sometimes impossible to measure. An individual (or his/her genotype) is *homozygous* at a certain site if (s)he has inherited identical alleles at that site. Otherwise the individual is *heterozygous* at that site. The concepts of haplotypes and genotypes are illustrated in Figure 1.2.

Haplotypes can be determined directly from *trios*. A trio consists of genotypes acquired from the individual and her (his) parents. However, trios require three times more genotyping and leave some sites without haplotype information. Even worse, sometimes it is not possible to measure genotypes of the individual's parents.

1.2 Background

The two problems studied in this thesis are the problem of *haplotype inference* and the problem of *local alignment significance*. Our results are presented in the four original articles included in this thesis. The solutions presented here use extensively *dynamic programming* and *Markovian models*.

In the haplotype inference problem (haplotyping) one tries to estimate the true haplotypes from genotypes by computational means. A genotype consists of combined information on a chromosome pair of an individual,

whereas the haplotypes have separate information on both chromosomes of the chromosome pair. The data consists of a sample of genotypes (trinary strings) from some population of individuals. For each genotype, an explaining haplotype pair (two binary strings) should be reported as accurately as possible.

In local alignment significance one computes the significance of the best pairwise local alignment score, a common measure of sequence similarity of two sequences x and y. Local alignment is a transformation of a substring of x into a substring of y. The best scoring local alignment finds the most similar substrings. If the score is high enough, it is possible that the substrings share some common biological function, e.g. a conserved gene. Significance is measured using the classical statistical framework, i.e. computing p-value. This p-value is the probability of picking two sequences from the null model that have as good or better best local alignment score than the sequences of interest. Thus, the problem is to compute a p-value from $(n, m, q_\cdot, t)$, where n and m are the lengths of the sequences x and y, $q_\cdot$ specifies the null model, and t is the local alignment score of interest.

Haplotypes contain information on which alleles of an individual are inherited together and therefore are important, e.g. when the genetic causes of a disease are studied. However, with current laboratory methods haplotypes are difficult to measure. On the other hand, genotypes are easier to measure. For this reason the problem of haplotype inference by computational means is also an important problem. Among our proposed solutions for haplotyping, the one in Paper III named BACH (Bayesian Context–based Haplotyping) is recommended as it is fast and gives good accuracy across different datasets.

The significance problem is very fundamental in bioinformatics. It is solved for example in BLAST (Basic Local Alignment Search Tool) [3], a well known and commonly used software for local alignment search. For example, the significance problem has an important role in homology search [52]. There, evolutionary related (homologous) genes are searched by looking for locally similar DNA or protein sequences (of genes). Local similarity is commonly measured by the best local alignment score and the p-value of this score helps to rule out sequences whose similarity is likely due to chance.

In Paper IV, a general framework is presented for the local alignment significance problem. This framework allows one to compute tight upper bounds for p-values in the presence of insertions and deletions (gaps). Unlike previous solutions for this significance problem, the method of Paper IV can handle gaps without troublesome sampling and curve fitting and does

not suffer from so–called edge effects. The framework computes its bound directly and exactly, whereas sampling produces a mean with some (standard) deviation.

1.3 Haplotype Inference

This section describes the haplotype inference problem solved in papers I–III.

1.3.1 Notation and Haplotype Inference Problem

Throughout this thesis, we will focus on SNP markers and assume that there are only two different values present at each SNP. We will denote the number of markers by m and the number of individuals (genotypes) by n. In Paper I, m and n have been used with opposite meanings.

The two alleles at each marker are denoted arbitrary as 0 and 1. Thus, a haplotype is a binary string listing the haplotype's alleles in the physical order in which they occur in the chromosome. A genotype is a ternary string of 0, 1 and 2, where these three values are $0 = \{0, 0\}$, $1 = \{1, 1\}$ and $2 = \{0, 1\}$. Thus, at each marker an unordered pair of alleles is listed in the physical order. This notation of haplotypes and genotypes is also illustrated in Figure 1.2.

Haplotypes $h = h_1 \cdot h_2 \cdots h_m$ and $h' = h'_1 \cdot h'_2 \cdots h'_m$ *explain* a genotype $g = g_1 \cdot g_2 \cdots g_m$, denoted as $g = \gamma(h, h')$, if either $h_i = h'_i = g_i$ or $g_i = 2$ and $h_i \neq h'_i$, for all $1 \leq i \leq m$.

The problem of *haplotype inference*, or *phasing*, is a problem of inferring haplotypes directly from genotypes. For a single genotype alone, one cannot differentiate between the $\max\{1, 2^{k-1}\}$ possible explaining haplotype pairs, where k is the number of heterozygous sites. For example, consider a genotype 222. The explaining haplotype pairs for this genotype are (000,111), (001,110), (010,101), and (011,100).

However, we can do better when we have a sample of related genotypes spanning the same sites. The common approach is to choose a model that explains the relations between haplotypes and then learn the model parameters from the genotype data. This approach has been used in Papers I, II, and III. Section 1.3.4 contains further discussion on why haplotype inference is generally possible.

Haplotypes are important true entities containing much of the genetic variation. They contain information on which alleles are inherited together and therefore they are the key–players when studying for example the genetic causes of a disease. However, with current laboratory methods they

are difficult to measure. For this reason the haplotype inference by computational means is also an important problem.

1.3.2 Accuracy of Haplotype Inference and Switch Distance

For some datasets the real haplotypes are known. These known haplotypes can be converted into genotypes from which haplotypes can be inferred. Based on these real haplotypes the accuracy of inferred haplotypes can be attained.

A common measure of accuracy is the *switch accuracy* [40]. It is based on the minimum number of switches needed to transform one haplotype pair into another (correct) one. A single *switch* exchanges equal length prefixes of the corresponding haplotypes. As an example, one switch transforms haplotype pair (0000, 1111) into (0011, 1100) and three switches are needed to get (0101, 1010) from the same pair (0000, 1111). Thus, the number of switches between (0000, 1111) and (0011, 1100) is 1 and the number of switches between (0101, 1010) and (0000, 1111) is 3. Note that the number of switches makes sense only if the genotypes explained by the haplotype pairs are the same.

The switch accuracy between two haplotype pairs is defined as $(k-1-s)/(k-1)$, where s is the minimum number of switches and k is the number of heterozygous positions (maximum number of switches is $k-1$). The switch accuracy is the proportion of equal (correct) switches in the haplotype pairs.

The number of switches s is a distance function (a metric) between two haplotype pairs, both explaining the same genotype. If the number of switches is normalized by $k-1$, we get the proportion of unequal (incorrect) switches in the haplotype pairs. Thus, this normalized distance equals 1 - switch accuracy. The experiments in Papers I, II, and III use either this normalized distance or the plain number of switches, both referred to as the *switch distance* (= switch error).

1.3.3 Methods for Haplotype Inference

The first attempt to solve the haplotype inference problem was Clark's algorithm [7] based on a single rule; from the already constructed haplotypes H, pick one h with $g = \gamma(h, h')$ for some genotype g not yet resolved. Then add h' to H, and mark as resolved all genotypes having an explaining pair of haplotypes in H. To begin, all genotypes that are heterozygous at most at one position, are marked as resolved and the corresponding haplotypes (these haplotypes are unambiguous) are added to the set of known haplotypes H. Then the rule is applied until all genotypes are resolved or the

process cannot be continued.

A heuristic strategy of using Clark's rule is to choose a solution that resolves most genotypes [7]. The problem of deciding whether all genotypes can be resolved with Clark's rule was later shown to be NP-hard [18].

Later, the problem of finding the smallest set of haplotypes explaining the genotypes was introduced in [20]. This *pure parsimony* problem has been shown to be NP-hard [37], but was often found to be easy to solve by integer linear programming [20].

A problem related to haplotype inference is to estimate the haplotype frequencies from genotypes. Two proposals [14] and [41] have been published relying on *Expectation–Maximization* (EM) algorithm to find a maximum likelihood solution to the haplotype frequencies. In this model, independent probabilities $p(h)$ are assigned to all possible haplotypes $h \in \{0,1\}^m$. Then the complete likelihood of the genotype data $G \subset \{0,1,2\}^m$ is defined as

$$\prod_{g \in G} \sum_{h,h' : \gamma(h,h') = g} p(h)p(h') \,. \tag{1.1}$$

From a maximum likelihood (or any) solution of Equation 1.1, the haplotype inference problem can be solved by picking haplotypes h and h' for each genotype g such that $\gamma(h,h') = g$ and h and h' maximize $p(h)p(h')$. Also the Gibbs sampling has been used to find these haplotype frequencies [63].

It is assumed in the model given by Equation 1.1 that the haplotypes of each genotype are independent of each other (*random mating*) and that the genotypes are independent of each other (individuals are not related).

The concept of *perfect phylogeny* was introduced to haplotype inference in [19]. In this approach, a tree describing the evolutionary history of the haplotypes is constructed. From a single haplotype (the root of the tree) the other haplotypes are formed by point mutations alone without recombination. It is assumed that there has not been recombination and that mutation occurs at most once at each position in the history (*infinite sites* assumption).

The haplotypes H are said to be in perfect phylogeny, if they can be built from a single haplotype by the process just described. The tree in Figure 1.3 illustrates perfect phylogeny. It can be decided in linear $O(mn)$ time, whether a set of genotypes can be explained by a set of haplotypes in perfect phylogeny [11]. However, when the genotypes (or even haplotypes) may have missing values, deciding whether there is a perfect phylogeny becomes NP-hard [61].

The concept of *imperfect phylogeny* was used for haplotype inference

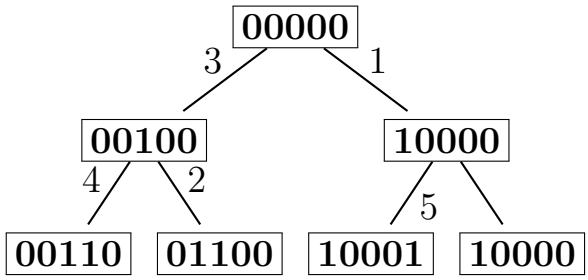

Figure 1.3: A perfect phylogeny for four haplotypes 00110, 01100, 10001, and 10000, shown as the leaves of the tree. Each node of this tree corresponds to a haplotype. The numbers on the edges denote the position that has been mutated, i.e. the nodes that are connected by an edge differ only at this position. It is assumed that each mutation occurs only once in the history (infinite sites assumption), so each number cannot occur more than once on the edges.

in software HAP [22], when the underlying haplotypes of genotypes do not strictly obey the perfect phylogeny. Imperfect phylogeny allows some mutations to occur multiple times in the tree–like history.

In the methods presented above, the physical locations of haplotype markers are not taken into account. Each marker is considered independently, thus the results obtained do not change if the order of the markers is shuffled. These methods are called not aware of the physical locations (NAPL), as opposite to methods aware of the physical locations (APL).

However, alleles at nearby markers are often correlated. This *genetic linkage* is due to the fact that when recombination is not present the alleles at the same chromosome copies are inherited together. The further away from each other two loci are the more likely recombination has occurred between them and less linked the alleles are at those loci. These correlations should be taken into account when larger chromosomal regions are considered.

The first solution to extend some (maybe NAPL) method X for larger chromosomal regions was the *partition ligation* technique [47]. First, the haplotype frequencies for the genotypes are inferred on short disjoint blocks of consecutive markers with method X. Some B haplotypes with highest frequencies are chosen and the solutions of adjacent blocks are merged to form B^2 haplotypes for the double–size blocks. Method X is used to get haplotype frequencies for the B^2 haplotypes and greedily B best of them are chosen. Continuing in this fashion, finally there will be only a single block containing a solution for the entire region.

Another way to extend NAPL methods for larger regions is based on the concept of *haplotype blocks* [10]. Haplotype blocks are regions of the genome, where the haplotype diversity is low compared to the length of the region. The relatively short regions between the blocks with high haplotype diversity are called *recombination hot spots*. For example the methods of [16, 34, 22] are based on haplotype blocks, i.e. they model nearby correlations by dividing the genotypes into blocks of consecutive markers. Then some method (maybe NAPL) is used to get haplotypes for the blocks and the transitions between the different haplotypes between adjacent blocks are modelled separately. As an example, the method of [34] uses Clark's [7] algorithm to get these haplotypes for a single block. Then the transitions between the blocks are modelled as a Markov model, i.e. the haplotype to be used in the next block depends only on the haplotype used in the current block (for each genotype).

The *block–free* methods, i.e. methods without block–structure assumption, have been found to infer haplotypes more accurately than the methods based on blocks (see experiments in [55, 58, 13]). The method HaploParser in Paper I is a novel generalization of the pure parsimony for longer chromosomal regions. It provides an unified solution to model long regions and genetic linkage with parsimony. This model can partition the haplotypes into blocks if the data suggests so. However, it is more flexible as it can partition the data into mosaic–like structure which can be different for each haplotype. This method is given a parameter K, the number of ancient founder haplotypes. When a sufficently large K is given, the optimal model is a pure parsimony solution. However, when K is smaller, it models haplotypes (and genotypes) with recombinations (linkage) and mutations in a block–free fashion. Moreover, the hierarchical parsing in HaploParser is a new and computationally feasible way to imitate ancient recombination graph model [25, 17] (see also the last paragraph of this subsection).

The method HIT in Paper II, based on hidden Markov models (HMM), is one alternative to model long haplotypes in a block–free way. A parameter K is given to these methods, acting as the number of founder haplotypes in HaploParser by affecting the model structure (topology). With a sufficently large K the optimal HMM is the maximum likelihood solution of Equation 1.1. With smaller K the HMM structure mimics recombinations (linkage) and also mutations. Novel and efficient algorithms are presented that compute the probability of a genotype given the HMM and learn these models from genotype data in EM–fashion. Thus, the method HIT is a novel generalization of the haplotype frequency estimation for longer chromosomal regions.

Method fastPHASE [58] is also based on the same topology as HIT. The main difference in fastPHASE compared to HIT is the transition probabilities; in fastPHASE there is only one transition parameter between adjacent markers (models the physical distance of markers) as in HIT there are independent transition probabilities between each state at adjacent markers. The model of HIT is biologically more plausible, as the recombination rate can vary between individuals, a fact caused by inversions [52]. A somewhat different method based on constrained hidden Markov models is given in [38].

In another direction, fast and accurate solutions using variable order Markov chains have been proposed for the haplotype inference problem in BACH (Paper III) and in HaploRec [13] and in Beagle [4]. The variable order Markov chains are well suited for modelling haplotype correlations over long regions, as they can capture high–order dependencies.

The HaploRec [13] is based on frequently occurring haplotype fragments. The parameters of its model are the frequencies of these haplotype fragments and they are learned by an EM–type algorithm. More recent Beagle [4] is based on efficient estimation of a related probabilistic automaton. This learned automaton specifies a single variable order Markov chain.

The method BACH of Paper III uses the Context Tree Weighting (CTW) algorithm to efficiently average over all variable order Markov chains (context trees) to obtain a Bayesian inference method for haplotyping. It uses maximum a posteriori (MAP) criterion for haplotype goodness. This criterion can be evaluated efficiently and exactly for any haplotype configuration. Method BACH achieves a robust behaviour as it does not need to learn model parameters or do model selection. The theory behind its model is very clean, heuristics are needed only in the algorithms exploring the posterior distribution (haplotypes). The accuracy obtained with BACH is very good over different datasets and it scales even to larger datasets.

The well–known program PHASE [63, 62, 39] is probably the most accurate haplotyping method (see experiments in Paper III and in [13, 55, 4]). It uses Bayesian Markov chain Monte Carlo methods to sample haplotypes from a distribution defined by a mosaic model. In this model, the haplotypes are constructed from fragments of the other haplotypes. A drawback of PHASE is that it is impractically slow on larger datasets.

The ultimate haplotyping method would take into account the true *coalescent* (history) of the haplotypes. A biologically faithful model to achieve this is the ancestral recombination graph [25, 17]. However, it turns out to be computationally infeasible. All the proposed methods are trade–offs between the accuracy and the computational feasibility.

1.3.4 Why Is It Possible to Infer Haplotypes?

Haplotype inference would be impossible, if the underlying haplotypes of genotypes were random strings distributed uniformly over all binary strings. However, this is not the case. Haplotypes are biological entities of individuals in a population.

Haplotyping is easier if the number of different haplotypes in the population is small. In the extreme case, only a single haplotype is present in the population. Then all individuals are homozygous at all markers and genotypes map to haplotypes uniquely. Typically there are more than one haplotype present, but there are factors limiting the variation in haplotypes.

The kinship of each individual limits the possible haplotypes (s)he can have, as each individual has her/his parents and parents' parents and so on. Thus, the haplotypes of an individual are conditional on her/his parents. Moreover, factors like like migration, genetic drift, and natural selection affect the population genetics [52] and reduce the number of different haplotypes.

As the haplotype diversity in the studied population is probably quite small, the natural criterion to use is parsimony, used for example in computational phylogenetics [52]. The pure parsimony haplotyping [20] is a direct application of parsimony, as its goal is to find the fewest haplotypes explaining the given genotypes. Parsimony is an application of the principle of Occam's razor. Occam's razor is "a scientific and philosophic rule that entities should not be multiplied unnecessarily which is interpreted as requiring that the simplest of competing theories be preferred to the more complex or that explanations of unknown phenomena be sought first in terms of known quantities" [44]. The last part of this definition can be seen in Clark's early haplotyping method [7].

The maximum likelihood (ML) haplotype frequency estimation [14, 41] can be seen as a probabilistic version of the pure parsimony criterion. The ML criterion is not as strict as the pure parsimony, but it still has the restriction that it cannot distribute the probability mass over too many haplotypes as then the likelihood would decrease.

If the genotypes span a large number of markers, it is likely that each genotype would require two unique haplotypes and then the pure parsimony and ML haplotype frequencies would lead to trivial solutions. However, this is the case that is solved in methods of HaploParser, HIT and BACH in Papers I, II, and III. All these methods take into account genetic linkage, i.e. the correlation of alleles at nearby markers. This linkage can be seen as (only) locally parsimonious haplotypes. Thus, the solutions of Papers I, II, and III use local parsimonia to infer plausible haplotypes. Moreover,

given a large enough parameter K, the optimal solutions of HaploParser and HIT converge to pure parsimony and ML haplotypes, respectively.

The method BACH uses Bayesian inference in haplotyping. The principle of simplicity or parsimonia can be found from the prior used to define its posterior probability. The prior probability for each context tree (variable order Markov chain) is the higher the simpler the model is. Moreover, these context trees can model genetic linkage very accurately. As BACH has the best overall accuracy in the experiments, it can be stated that haplotyping is possible by modelling linkage by simple models.

1.4 Local Alignment Significance

This section describes the problem solved in paper IV – deciding whether local similarities in two sequences occur because the sequences are related or just by chance. A local alignment score is used as the similarity measure of sequences and the classical statistical hypothesis testing, i.e. p-values are used to access the significance of the best local alignment score.

1.4.1 Alignment Scoring Model

The basic processes that alter biological sequences are mutation and selection. In this chapter, the basic mutation events are *substitutions*, *deletions*, and *insertions*. A substitution event changes a single residue from the sequence, whereas one or more residues are deleted and inserted in deletion and insertion events. The insertions and deletions are referred to simply as *gaps* or *indels*.

Let $x = x_1, x_2 \ldots, x_n$ and $y = y_1, y_2, \ldots, y_m$ be the two sequences from a finite set of residues Σ. The alignment scoring model uses basic mutation events to transform sequence x to sequence y. Each mutation event has an associated *score*. The model is additive, i.e. the score of transforming x to y is the sum of single event scores. Each substitution that changes residue a to residue b has a score $s(a, b)$. The *linear gap model* is assumed, where the score of a single residue deletion and a single residue insertion has an additive score of $-d$ or a *cost* of d.

Any transformation of x to y is a *global alignment*. A *local alignment* corresponds to a similar transformation, where only substrings of x and y are transformed. From now on, only local alignments are considered.

A local (and global) alignment can be described as a path in the *alignment grid* (i, j), where $i = 0, \ldots, n$ and $j = 0, \ldots, m$. This path consists of three kinds of steps (edges), vertical, diagonal, and horizontal steps. A diagonal step from (i, j) to $(i + 1, j + 1)$ corresponds to aligning residues

x_{i+1} and y_{j+1} and has a score of $s(x_{i+1}, y_{j+1})$, a horizontal step from (i, j) to $(i+1, j)$ deletes character x_{i+1} and has a score of $-d$, and a vertical step from (i, j) to $(i, j+1)$ inserts character y_{j+1} and has a score of $-d$. The total score of an alignment is the sum of the scores of individual steps.

As an example, consider the alignment of ATCGCT and GACGGT:

```
A-TCG
ACG-G
```

This alignment has two gaps (one insertion and one deletion) and a score of $s(A, A) - d + s(T, G) - d + s(G, G)$. The alignment is shown in the alignment grid in Figure 1.4.

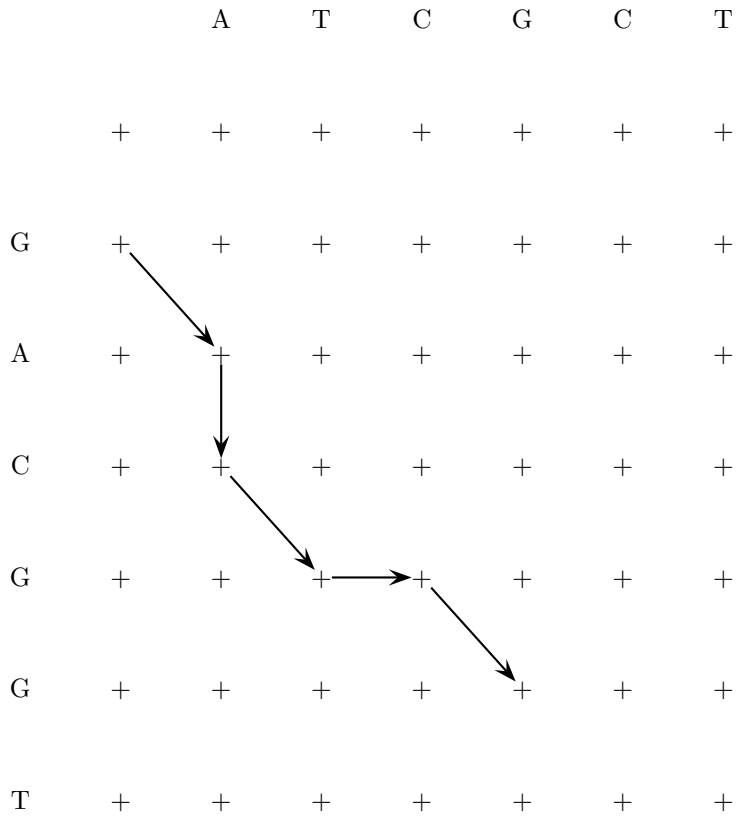

Figure 1.4: The alignment grid and an example alignment of ATCGCT and GACGGT.

Typically one is interested in the best scoring alignment of the given sequences. In the next subsection, the well–known Smith–Waterman algo-

rithm [59, 15] is described to efficiently compute the best local alignment score of a sequence pair (x, y).

Smith–Waterman Algorithm

The Smith–Waterman Algorithm [59, 15] computes the best local alignment score of sequences of length n and m in time $O(mn)$.

The algorithm uses dynamic programming to compute table H, where each element $H(i, j)$ states the best local alignment score (of x and y) ending at (i, j) in the alignment grid, or equivalently the best alignment score of suffixes of $x_1 \cdots x_i$ and $y_1 \cdots y_j$.

By setting $H(0, 0) = H(0, j) = H(i, 0) = 0$ the dynamic programming of H becomes

$$H(i,j) = \max \begin{cases} 0, \\ H(i-1, j-1) + s(a_i, b_j), \\ H(i-1, j) - d, \\ H(i, j-1) - d\,, \end{cases} \tag{1.2}$$

where $i = 1, \ldots, n$ and $j = 1, \ldots, m$. The best local alignment score S is now the maximum $H(i, j)$ over all i, j, i.e. $S = \max\{H(i, j) \mid i = 0, \ldots, n,\ j = 0, \ldots, m\}$. The substrings of x and y corresponding to the local alignment with score $H(i, j)$ can be found by a standard trace–back on table H starting from $H(i, j)$. This trace–back finds the path in the alignment grid describing the corresponding alignment.

1.4.2 Statistical Significance of a Local Alignment Score

Having computed the best local alignment scores t for some sequences x and y, a natural question is how significant this alignment is. To answer this question, the classical statistical framework is used to compute the *p-value*, referred to as p_t. The value of p_t is the probability of getting a sequence pair (x', y') from the *null model* with best local alignment score $\geq t$.

The null model used here is a simple random *Bernoulli model* [12]. In this model, the probability of sequence x is given as $P(x) = \prod_{i=1}^{n} q_{x_i}$, where q_a is the probability of residue a. A similar probability is assigned to sequence y. The Bernoulli model does not model the length distribution of the sequences as it computes a proper probability distribution only for fixed–length sequences. Thus, the sequences x' and y' from the null model are of fixed lengths n and m, respectively.

1.4.3 Methods for Computing the p-value

An estimation $\hat{p_t}$ of the p-value p_t can be obtained by a simple Monte Carlo method [23]. By sampling N sequence pairs from the null model, the value of $\hat{p_t}$ is the fraction of sampled sequence pairs that have the best local alignment score $\geq t$. This sampling procedure has a standard deviation of $\sqrt{p_t(1-p_t)/N}$ [23, 48, 46].

This simple Monte Carlo method is not very practical if small p-values are needed. The sample size N must be larger than $1/p_t$ in hope to get at least a single positive case to the sample (a sequence pair with local alignment score $\geq t$). Importance sampling [23] has been used to sample small p-values with fewer samples [46, 24]. There the sequence pairs are sampled from a distribution that gives higher probability for those rare sequence pairs with a high best local alignment score. By weighting these samples properly, an estimate of p_t is obtained.

The most widely used method to approximate the significance of the alignment score is to use the *Karlin–Altschul statistics* [29, 30]. There the significance of alignments *without gaps* is approximated as a one–dimensional problem. A single string of length mn is created from the alignment problem (two–dimensional problem in the alignment grid). Strings x' and y' from the null model are aligned (globally) in all $m+n-1$ different ways and the resulting string pairs are put together to form a string $Y = Y_1, \ldots, Y_{mn}$ of mn letter pairs ($Y_i \in \Sigma \times \Sigma$). Then a probability $q_a q_b$ is assigned for each letter pair (a, b). This dimensional reduction can be seen as the concatenation of the diagonals of the alignment grid (but skipping the first row and column) to get a single linear chain of nodes.

The best one–dimensional local alignment score of Y is the *maximal segment score* $M(Y)$:

$$M(Y) = \max_{1 \leq i \leq j \leq mn} \sum_{k=i}^{j} s(Y_k)\,, \tag{1.3}$$

where $s(Y_k = \{a, b\})$ is $s(a, b)$.

This dimension reduction does not give an exact solution for the original two–dimensional problem because of the so–called *edge effects*; some alignments can overlap the concatenation points, i.e. these alignments do not correspond to any real alignments. Moreover, the mn letter pairs of Y are not independent, as they depend on the two underlying strings x' and y'. In fact, if a string of mn letter pairs is positionally independent, there are $|\Sigma|^{2mn}$ different strings of independent mn pairs of letters, while there are only $|\Sigma|^{m+n}$ different Ys as this number equals the number of different string pairs (x', y').

In Karlin–Altschul statistics the probability that the maximum alignment score $M(Y)$ is greater than t is approximated as an extreme value distribution (Gumbel) given by

$$P(M(Y) > t) \approx 1 - \exp(-Kmne^{-\lambda t}), \tag{1.4}$$

for some parameters K and λ. Equation 1.4 is based on assumptions that there are no gaps, the expected score $\sum_{a,b} q_a q_b s(a,b)$ is negative, and that for some letter pair (a,b) $(q_a q_b > 0)$ $s(a,b) > 0$. With these assumptions the parameters K and λ can be solved analytically [29, 28]. There are also ways to reduce the edge effects [2].

There is empirical evidence that the distribution of the best alignment scores *with gaps* follow the same distribution quite accurately, e.g. [51]. However, no general analytical solution to find K and λ is known with gaps. Some efficient solutions [5, 6, 53] are presented to fit the parameters λ and K to gapped alignment scores.

The distribution of $M(Y)$ can also be computed exactly by the method of Mercier et al. [43], and by assuming that all scores are integers. This solution is based on the Markov chain illustrated in Figure 1.5.

There is a state in this chain for each alignment score $0, 1, \ldots, t$ and the transition probability from state $i = 1, \ldots, t-1$ to state $j = 1, \ldots, t-1$ is $P(s(a,b) = j - i) = \sum_{a,b \in \Sigma : s(a,b) = j-i} q_a q_b$. The transition probability from state $i = 0, \ldots, t-1$ to state 0 is given as $P(s(a,b) \leq i)$, and the transition probability from state $i = 0, \ldots, t-1$ to state t is given as $P(s(a,b) \geq t - i)$. The only transition from state t is to state t with probability 1.

The probability of $P(M(Y) \geq t)$ is the probability of this Markov chain to be in state t after mn steps started from state 0. This probability is P_{0t}^{mn}, where P is the transition matrix of this chain, i.e. P_{ij} is the transition probability from state i to j and P^{mn} is the mnth power of P. This method has a pseudo–polynomial time complexity (on the score values) and with similar assumptions as made in Paper IV the time complexity is $O(\min\{mnt, t^{2.376} \log(mn)\})$.

In Paper IV, a general framework is presented for the local alignment significance problem. This framework allows one to compute tight upper bounds for p-values in the presence of insertions and deletions (gaps). Having an upper bound is an advantage compared to the previous methods that compute an approximate p-value, which can be larger or smaller than the correct one. Typical solutions based on sampling, produce only a mean and a standard deviation of the p-value. Unlike most previous solutions for this significance problem, the method of Paper IV does not use any sampling and curve fitting and does not suffer from so–called edge effects. The

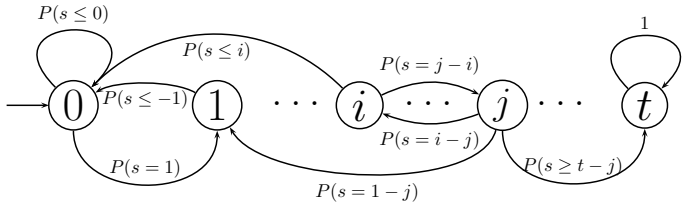

Figure 1.5: A Markov chain used in [43] to compute $P(M(Y) \geq t)$. The states are numbered according to the alignment scores. For clarity, all transitions have not been drawn. Term $P(s = c)$ is used to denote probability $P(s(a, b) = c)$ (and similarly for $P(s \leq c)$ and $P(s \geq c)$). This figure is from Paper IV.

algorithm of Paper IV computes this upper bound directly and exactly, in a similar fashion as the method of [43] does without gaps. The algorithm have a pseudo–polynomial time complexity and for typical instances it is fast (polynomial).

1.5 Main Contributions with Short Summaries

Papers I–IV constitute the core of this thesis. The main research contributions in these papers are:

Paper I: A method called HaploParser is presented for the haplotype inference problem. HaploParser uses a combinatorial mosaic model of recombinations and point mutations. Algorithms based on dynamic programming and on greedy heuristics are introduced to parse unphased genotypes in terms of (implicit) founder haplotypes in a flat and in a hierarchical fashion. As a by–product of the parse, the haplotypes are inferred for the given genotypes.

Paper II: A method called HIT – Haplotype Inference Technique – is presented for the haplotype inference problem. HIT models haplotypes using a hidden Markov model (HMM) with a special topology that mimics recombinations and point mutations. A closed form EM algorithm, among other algorithms, is presented for the learning of these HMMs from unphased genotype data. From this learned model the haplotypes can be inferred. An extended version of Paper II has been published in [55].

Paper III: A method called BACH – Bayesian Context-based Haplotyping – is presented for the haplotype inference problem. In BACH,

the Markov model is of variable order while in paper II it was of fixed order 1. The context tree weighting algorithm is utilized to efficiently compute the weighted average over all variable order Markov chains to evaluate the posterior probability (goodness) of a haplotype configuration. Algorithms are presented that use Bayesian maximum a posteriori (MAP) criterion to find haplotypes for a given set of genotypes.

Paper IV: A general framework is presented to compute a tight upper bound for the p-value (significance) of a pairwise local alignment score. Unlike the previous solutions for this significance computation, the new framework handles alignments with gaps without troublesome sampling and curve fitting and does not suffer from so–called edge–effects. The algorithms in this framework have pseudo–polynomial time complexities, and for typical instances they are fast.

The author has made a major contribution to all of the included papers. The work with the haplotype inference problem started from the author's MSc thesis that generalized the model of [66] for genotypes and used it for haplotyping. The thesis included the flat parsing of genotypes described in Paper I.

After this preliminary work, the author added transition probabilities between adjacent markers of the founder sequences to this combinatorial model. This model was a hidden Markov model with each state emitting only the allele corresponding to the founder allele. The founders were first found by minimizing the flat parsing score and then an EM–type algorithm was used to find maximum likelihood transition parameters. The switch accuracy with this model was very good as being comparable to the accuracy of PHASE [63].

When the news of good results and the pictures of this model found their way to Mikko Koivisto and Esko Ukkonen, Mikko urged the author to add emission parameters to the HMM to get rid of the combinatorial model in the beginning. The idea was to use the EM algorithm to learn the emission parameters, as well. With some thinking, the author implemented the closed form EM algorithm explained in Paper II. Later, it was noticed that in [32] an EM–type algorithm was used for a special case of this problem and this algorithm was not in closed form as it resorted to a numerical solver. Because of this, we gave our EM algorithm a closer look and finally Mikko proved that this algorithm converges and it can be derived by adding genotype phase information to the hidden data of the EM algorithm. By fixing some additional details the method HIT was introduced in paper II

in 2005. Later the same year, the method called fastPHASE [58] based on a similar hidden Markov model was introduced (citing Paper II).

Paper I was written later, including the new idea of hierarchical parsing. This hierarchical parsing is interesting because it is very close to constructing a minimal Ancestral Recombination Graph (ARG) for the input genotypes.

Mikko Koivisto and Jussi Kollin introduced the context tree weighting to the author who fixed the details on how to use it in haplotype inference with help from Mikko and Jussi.

Paper IV is a by–product of the author's random walk in the pattern matching algorithms. The idea was to add sequences as distributions to the algorithm counting the number of suboptimal alignments [45]. Then it was just the matter of figuring out what the framework did and how to use it for something useful. Fortunately, Kimmo Palin had introduced the problem of computing the p-value of a local alignment score to the author [48].

The rest of this thesis is organized as follows. First the methods of HaploParser (Paper I), HIT (Paper II), and BACH (Paper III) are explained in Chapters 2, 3, and 4, respectively. Chapter 5 sketches the novel approach of Paper IV for local alignment significance. And finally, Chapter 6 concludes this thesis.

Chapter 2

A Combinatorial Mosaic Model for Haplotype Inference

In this chapter, a haplotype inference method called HaploParser from Paper I is presented. It models haplotypes and genotypes with recombinations and point mutations using a combinatorial mosaic model. The model of HaploParser is a generalization of [66], as of modelling genotype data and point mutations. From this model the haplotypes inference research venture started, leading to this thesis. Later, this model was extended to a hidden Markov model based solution HIT in Paper II, which will be presented in the next chapter. The algorithms of HaploParser share many elements with the ones used in HIT.

2.1 Mosaic Model of HaploParser

The underlying model of HaploParser assumes that the current population is evolved from a small number of 'founder' individuals by recombinations and by some mutations. Hence, the haplotypes of the current population are recombinations of the *founder haplotypes*, i.e. the haplotypes of the founder individuals.

A *parse* of haplotypes describes which haplotype alleles are inherited from which founder haplotype. Thus, the parse describes the evolutionary history of the haplotypes. If each founder haplotype is given a distinct *color*, then a parse defines a *coloring* for the current haplotypes. This coloring reveals a *mosaic*–like structure of haplotypes. As the coloring is defined for the haplotypes, genotypes can be colored by having a coloring on explaining haplotypes of each genotype.

Modelled recombinations can be spotted from this mosaic structure as

a position where the color changes along a haplotype. In Figure 2.1, three founder haplotypes define the coloring of the haplotypes. The number of color changes in this figure is 13, thus 13 recombination events are modelled. To model point mutations, there could be some mismatches between haplotype alleles with color k and the corresponding alleles of the founder haplotype with the same color k.

2.2 Parsing

The term *parsing* means getting a coloring for the genotypes (or haplotypes) with respect to some founder haplotypes. In Paper I, flat and hierarchical parsing were used. In both of them a parse is chosen that minimizes a particular score. This score is the number of recombination events plus c times the number of point mutations, where c is a given parameter. The parse of a genotype fixes its two haplotypes, thus parsing can be used for haplotype inference.

For the time being, we assume that appropriate founder set F containing K haplotypes is known. Moreover, all the genotypes of G and the haplotypes of F are of length m. Next we present the flat and the hierarchical parsing of genotypes G with respect to F.

2.2.1 Flat Parsing

In flat parsing one uses segments of the sequences F to construct a pair of haplotypes for each genotype. As haplotypes F are defined on the same markers as the data G, each segment can only be used at the same position as it occurs in F. Thus, a new haplotype from F is constructed by choosing alleles at each marker from some $f \in F$ at the corresponding marker. However, to make the problem interesting, we want to minimize the number of positions where corresponding founder haplotypes f are changed. This minimization is done independently for (both haplotypes of) each genotype.

The point mutations have been incorporated into the model as follows. A parameter $c > 0$ is chosen as a cost of changing a single allele in the parse, and then a cost of 1 is charged for each color change. Then the score to minimize is the sum individual costs, i.e. the number of color changes plus c times the number of allele mismatches. The minimum of such a score is denoted as $score^c_F(G)$ for a set of genotypes G.

The mutation part of the model is not used in the experiments of Paper I, as the parameter c was set to the high value of 100. However, allowing mutations in the parse simplifies the computations, as then any haplotype can be parsed with any non–empty set of founders.

Each genotype g ($\in G$) can be parsed independently, given the set of founders F. The minimum score, $score_F^c(g)$, can be computed using dynamic programming.

Let $S(i, a, b)$ be the minimum score of partial genotype $g_1, \cdots, g_i$ when g_i is parsed from founder sequences a and b (alleles F_{ai} and F_{bi} explain g_i). The value of $S(i, a, b)$ can be computed as follows.

$$\begin{aligned} S(0, a, b) &= 0 \\ S(i, a, b) &= p_c(g_i, F_{ai}, F_{bi}) + \min_{a',b'} \left(S(i-1, a', b') + I_{a' \neq a} + I_{b' \neq b} \right), \end{aligned} \tag{2.1}$$

for $a, b = 1, \ldots, K$, and $i = 1, \ldots, m$. Here K is the size of F ($= |F|$), I_A is the indicator of a predicate A, i.e.

$$I_A = \begin{cases} 1 & \text{, if } A \text{ is true} \\ 0 & \text{, othewise}, \end{cases}$$

and $p_c(t, s, s')$ is the cost of mutating genotype t (of length 1) to $\gamma(s, s')$, i.e.

$$p_c(t, s, s') = \begin{cases} 0 & \text{, if } t = \gamma(s, s') \\ 2c & \text{, if } t \neq \gamma(s, s'') \text{ and } t \neq \gamma(s'', s') \text{ for all } s'' \in \{0, 1\} \\ c & \text{, otherwise.} \end{cases}$$

Moreover, term g_i is the allele of g at ith marker and F_{ai} is the allele of founder sequence a at marker i.

The minimum score, $score_F^c(g)$ is $\min_{a,b} S(m, a, b)$ and the parse can be found by a standard trace–back. The trace–back gives for each $i = 1, \ldots, m$ one $S(i, a_i, b_i)$ and the parse uses at marker i founders a_i and b_i (two paths, one corresponding to a:s and the other to b:s). Thus, the two haplotypes with a score of $score_F^c(g)$ together can be constructed by concatenating founder alleles $F_{a_i i}$ and $F_{b_i i}$ separately for each $i = 1, \ldots, n$.

The time complexity of directly evaluating Equation 2.1 is $O(mK^4)$, but by clever evaluation it can be done in $O(mK^2)$ time as shown in Paper I. Note that the parse found by a trace–back suggests a haplotype pair for the genotype g. Thus, it is straightforward to use it for haplotype inference.

2.2.2 Hierarchical Parsing

The flat parsing does not take into account the coalescent of the sequences, as each genotype is parsed independently. For example, if some specific color change is used for parsing almost all m input sequences, the score is

increased by almost m. It is more plausible that there has been a single recombination early in the history and individuals of the current population are descendants of the individual with this recombination. So the score should increase only by one!

To get a biologically more plausible combinatorial model, hierarchical score $hscore^c_F(G)$ is defined. This score is best explained via a procedure in which one picks two founder sequences f_1 and f_2 at a time, and then adds a combination of length n of a prefix of f_1 and a suffix of f_2 to the founder set F. The idea is to explain the genotypes with a minimum number of prefix–suffix combinations, i.e. *recombination events*. Thus, $hscore^c_F(G)$ is the minimum number of these recombination events needed to explain all genotypes G when the procedure is started from founders F.

An efficient algorithm is not known for the problem of minimizing $hscore^c_F(G)$. Our heuristic algorithm uses $score^c_F(G)$ in a greedy fashion, i.e. it picks a combination that decreases $score^c_F(G)$ the most at every iteration. This process is iterated until all genotypes can be explained from the founder set or when a predefined number of iterations have been reached.

The hierarchical parsing can be seen as constructing a graph similar to an ancestral recombination graph (ARG) [25]. If we would add point–mutations in the same iterative manner as recombinations, we could start the hierarchical parsing from a single sequence (common ancestor). Then added mutations would correspond to mutation edges and added recombinations to cycles in the graph, thus a typical recombination graph would be constructed. An infinite sites assumption could be enforced by allowing only a single mutation to be used at each position i. Then the minimization of recombinations would lead to the minimal ancestral recombination graph [25, 17, 60]. Figure 2.2 gives an example of the hierarchical parsing.

2.3 Finding a Founder Set for Flat Parsing

The inverse problem of finding a set F that minimizes $score^c_F(G)$ for a set of genotypes G is NP-hard as shown in Paper I, but can be solved in polynomial time in some special cases with a founder set of size 2 [Paper I] and [66, 69]. In case of haplotype data, there have been some attempts to solve this problem exactly [66, 69].

The algorithm in Paper I constructs each column of F from left to right in a greedy fashion. When column i is decided, each of columns $1, \ldots i-1$ are kept fixed and the alleles at column i minimizing $score^c_F(G)$ up to column i are chosen. After the first greedy construction round, another round from left to right is made but now both sides of the current column

are taken into account to the score.

The time complexity of this algorithm is $O(mnK^2 2^K)$ for a dataset of n genotypes over m SNPs. The algorithm finds the optimal solution when $K = 2$ and $c = \infty$.

2.4 Finding a Founder Set for Hierarchical Parsing

To start the hierarchical parsing, we need at least two founder sequences. In Paper I, we did not try to find the best of such founders. Instead, the heuristic algorithm to construct a founder set for flat parsing was used. As the problem of minimizing $score^c_F(G)$ can be found efficiently in the case of two founders ($K = 2$), this would be an ideal solution to start from. However, the experiments in Paper I show that the hierarchical parsing gives more accurate results if $K > 2$.

2.5 HaploParser in a Nutshell

The haplotype inference method HaploParser is the following for the given genotypes G and the parameters K and k.

- Find founder haplotypes F_0 minimizing $score^c_{F_0}(G)$.
- Find founder haplotypes $F_1, F_2, \ldots F_k$ by the greedy hierarchical parsing algorithm.
- Parse genotypes using F_k and output haplotypes inferred by the parse.

In the experiments of Paper I, the parameter k was either 0 (no hierarchical parsing) or it was the smallest value for which the hierarchical parsing decreased the score by at most 1, i.e. $k := min\{k' | score^c_{F_{k'}}(G) - score^c_{F_{k'+1}}(G) \leq 1\}$. The latter criterion was argued by the fact that if the score decreases by one, there is no difference between the flat and the hierarchical parsing.

2.6 Experimental Results

In Paper I, 220 datasets were obtained from the HapMap database [65]. Each of these datasets consists of 120 haplotypes over 100 SNPs. These haplotypes were converted into 60 genotypes, and the switch distance was used to measure the error in the inferred haplotypes.

The results with HaploParser were on average best with $K = 10$ founders (not always as shown in Figure 2.3) and when hierarchical parsing was used. However, the results were about 30% worse than with the HMM–based methods HIT (of Paper II) and fastPHASE [58]. The most interesting result was the effect of hierarchical parsing. It improved the result consistently for almost all datasets. This effect is shown for a single dataset in Figure 2.3. The x–axis in this figure is the number of greedy iterations with the hierarchical parsing algorithm and the y–axis is the (unnormalized) switch distance. On average, this hierarchical parsing improved the results by 30%. The runtime of HaploParser's Java implementation was a couple of minutes on a single HapMap dataset (100 SNPs) on a standard desktop PC. However, the runtime on larger datasets might be too high as the runtime of hierarchical parsing is superlinear on the data size mn.

HaploParser is not included in the experiments of the following chapters, as its average haplotyping accuracy does reach the accuracy of HIT (which is included).

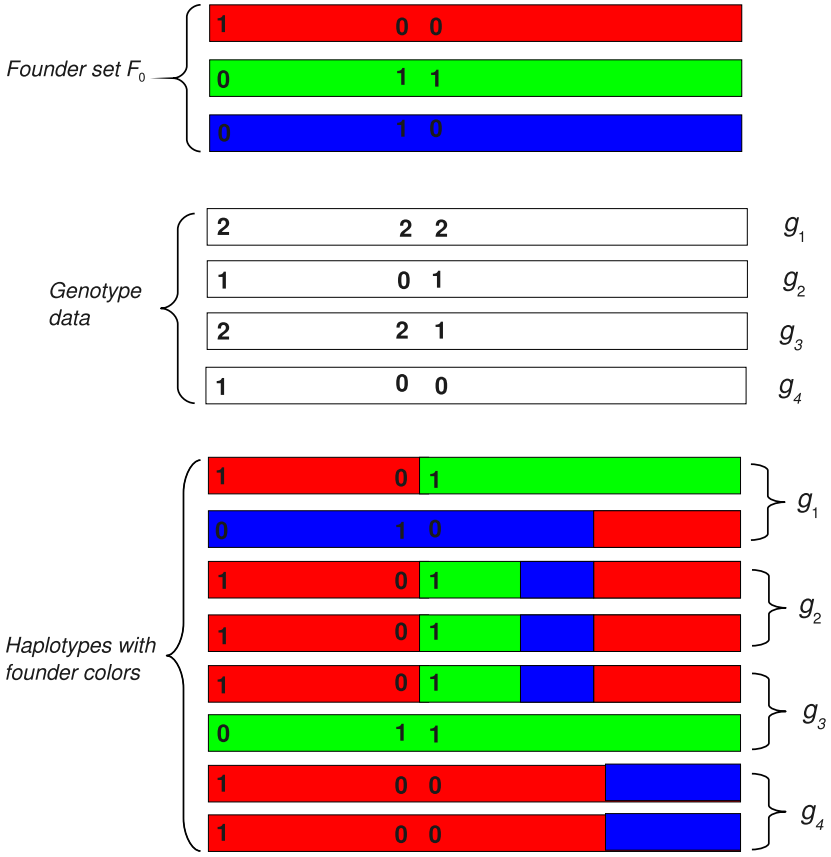

Figure 2.1: An example of flat parse for four genotypes (middle). For each genotype there is an explaining pair of haplotypes (down) parsed from founder haplotypes (top). The score of this parse is 13 (assuming there are no mutations).

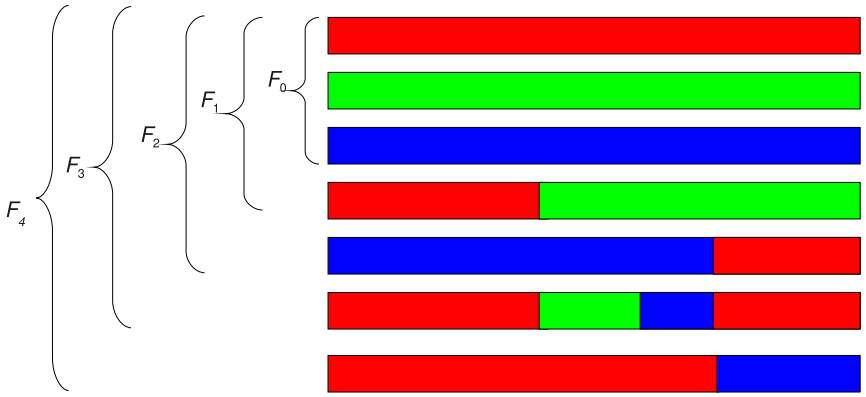

Figure 2.2: An example of hierarchical parsing. The founder haplotypes (three topmost) are the same as in Figure 2.1. At each iteration i, a new founder haplotype f is added to the founder set which is denoted by $F_i = F_{i-1} \cup \{f\}$. This new haplotype f is a recombinant of two haplotypes in F_{i-1}. The hierarchical score in this case is 4, as F_4 contains the data (all color patterns in the Figure 2.1). Note that the flat score is 13 (Figure 2.1).

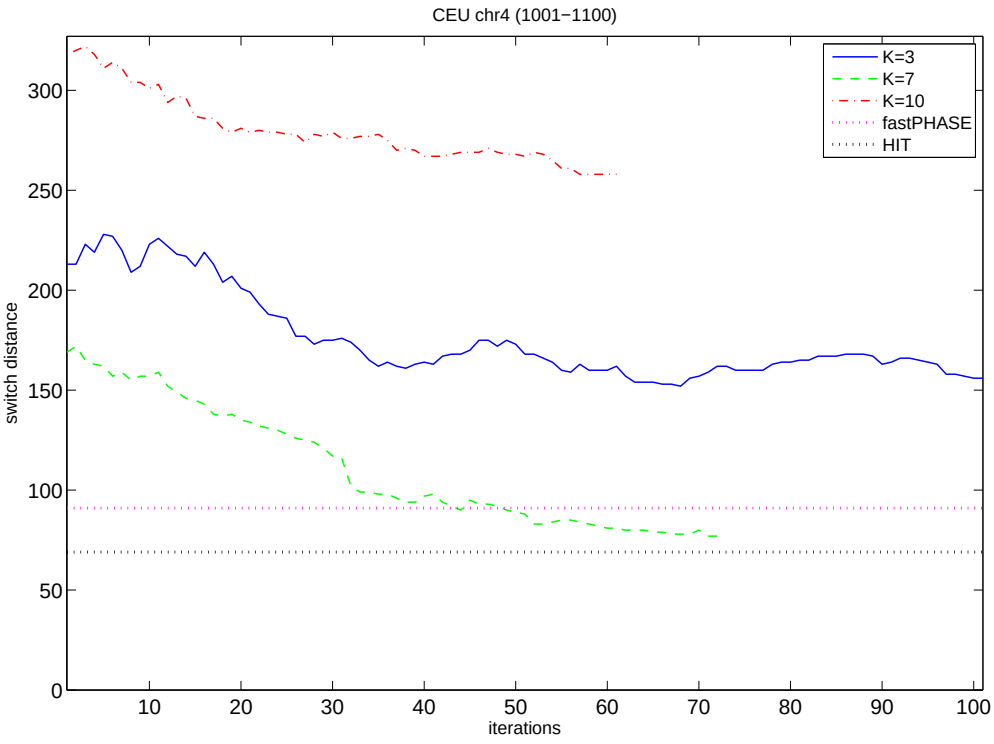

Figure 2.3: The effect of hierarchical parsing, when the greedy algorithm is started from $K = 3, 7, 10$ initial founder haplotypes. This plot is obtained by plotting the switch distance after each greedy iteration of the hierarchical parsing algorithm. Surprisingly, the increase from $K = 7$ to $K = 10$ decreases the accuracy. The horizontal lines show the switch distance obtained with HIT [55] and with fastPHASE [58].

Chapter 3

A Hidden Markov Model for Haplotype Inference

In this chapter, a hidden Markov model (HMM) based method called HIT (Haplotype Inference Technique) for haplotype inference is presented. The HMM used in HIT has a special structure that resembles the combinatorial founder model of HaploParser. The standard algorithms to learn HMMs do not apply here, because the learning should be done from data consisting of genotypes. We assume that that a reader is familiar with hidden Markov models and their typical use in biological sequence analysis (See e.g. [12, 54]).

3.1 Model Structure of HIT

The hidden Markov model structure used in HIT is best described using an example shown in Figure 3.1. At each marker there are K states, each state having an emission distribution for alleles 0 and 1. Then there is a single begin state at imaginary marker 0 which do not emit any alleles. Moreover, there are transitions between states at adjacent markers. The model structure is fully determined by the parameter K and the length of genotypes m. In total, there are $mK+1$ states and $K+(m-1)K^2$ possible transitions.

The parameters of this model are the transition probabilities $\tau(a, b)$ between states a and b and the emission probabilities of emitting an allele x from state a, $\epsilon(x, a)$. Moreover, each path from the begin state to one of the rightmost states emits a haplotype of length m.

This model structure has been (independently) proposed and used for disease association [33], as well as for genotype error detection [31], and for

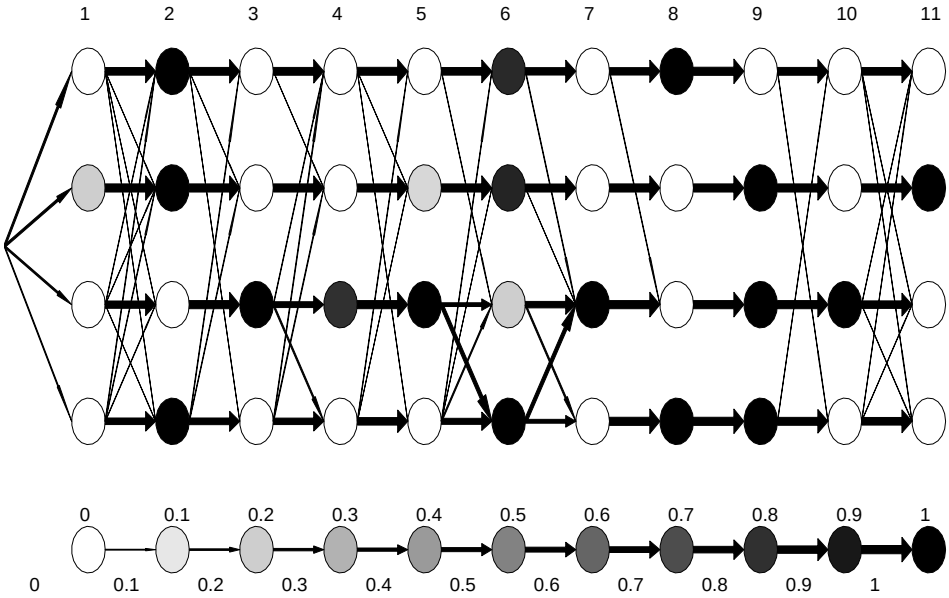

Figure 3.1: An example of HMM structure of HIT for SNP sequences of length 11 with $K = 4$ founders. The color intensities and the arrow widths reveal the emission and transition probabilities. This figure is from [55].

local ancestry inference [49].

Using the basic *forward algorithm* [12], one could compute the probability $P(h|M)$ of a haplotype h given the HMM $M = (\tau, \epsilon)$. Next we show how to compute the probability $P(g|M) = \sum_{h,h':\gamma(h,h')=g} p(h, h'|M)$ of a genotype g (term $p(h, h'|M) = p(h'|M)p(h|M)$).

3.2 Forward Algorithm for Genotypes

The probability $P(g|M)$ of a genotype g given the HMM M can be computed by dynamic programming similar to Equation 2.1. Let a_j and b_j be two states of the HMM at marker j and denote by $L(a_j, b_j)$ the probability of partial genotype $g_1, \ldots, g_j$ when the allele g_j is emitted from states a_j and b_j. The dynamic programming of $L(a_j, b_j)$ is as follows.

$$\begin{aligned} L(a_0, b_0) &= 1 \\ L(a_{j+1}, b_{j+1}) &= P(g_{j+1}|a_{j+1}, b_{j+1}) \sum_{a_j, b_j} L(a_j, b_j)\tau(a_j, a_{j+1})\tau(b_j, b_{j+1}) \,, \end{aligned} \tag{3.1}$$

where $\tau(a_j, a_{j+1})$ is the transition probability (parameter) from state a_j to state a_{j+1} and $P(g_j|a_j, b_j)$ is the probability of emitting an allele g_j from states a_j and b_j, i.e.

$$P(g_i|a_j, b_j) = \begin{cases} \epsilon(0, a_j)\epsilon(0, b_j) & \text{, if } g_i = 0 \\ \epsilon(1, a_j)\epsilon(1, b_j) & \text{, if } g_i = 1 \\ \epsilon(0, a_j)\epsilon(1, b_j) + \epsilon(1, a_j)\epsilon(0, b_j) & \text{, if } g_i = 2\,, \end{cases}$$

where $\epsilon(x, a_j)$ is the emission probability of x from state a_j.

The probability $P(g|M)$ is $\sum_{a_m, b_m} L(a_m, b_m)$. Direct evaluation of Equation 3.1 takes $O(mK^4)$ time, but a faster $O(mK^3)$ time complexity can be achieved as shown in Paper II. Using the fastest known matrix multiplication algorithm [8], the time complexity can be further improved to $O(mK^{2.376})$.

3.3 Model Training with an EM Algorithm

The maximum likelihood (ML) principle was used to find the parameter values for the hidden Markov model M of HIT. Thus, the parameters should be set to maximize the likelihood of the data according to Equation 1.1, i.e. maximize $P(G|M) = \prod_{g \in G} P(g|M)$, where each term $P(g|M) = \sum_{h,h':\gamma(h,h')=g} p(h, h'|M)$ is computed as shown in Section 3.2.

In Paper II, the commonly used EM algorithm was used to find these parameters. This is because the exact solution is hard to compute and even if it would be computable, typically the ML solution is not the biologically most plausible one. Paper II found that the correlation between likelihood and haplotype prediction accuracy is not very high. In the experiments of Paper II, Paper III, and [55, 57], it has been observed that it is better to combine several solutions with high likelihood, than to pick the solution with the highest (found) likelihood.

The EM algorithm starts from some initial set of parameters $\theta^{(0)}$ and iteratively improves the current parameters $\theta^{(r)}$ by equation

$$\theta^{(r+1)} := \arg\max_{\theta} \sum_{Z} P(Z \mid G, \theta^{(r)}) \ln P(G, Z \mid \theta) \tag{3.2}$$

where Z runs through a chosen additional hidden data and G is the set of genotypes (data). In the Baum–Welch algorithm [12, 54] used to learn HMMs from single (haploid) sequences, the hidden data Z consists of a path through the model. However, with genotype data two paths for every genotype is not sufficient hidden data Z. The hidden data Z used in HIT consists of T containing two paths for each genotype through the model

and the genotype phase information U. The hidden data U describes the two haplotypes for each genotype.

With $Z = (T, U)$, the term $P(G, Z \mid \theta)$ of Equation 3.2 simplifies into simple products. Moreover, taking ln of the term $P(G, Z \mid \theta)$ separates the emission and transition parameters, so they can be estimated separately. Using standard techniques from constrained optimization, one gets intuitive updating formulas for the emission and transition parameters. That is, the transition parameter from state a to state b in $\theta^{(r+1)}$ is proportional to the expected number of transitions from state a to state b (with parameters θ^r). Similarly, the parameter corresponding to an emission of allele y from state b is proportional to the expected number of emissions of y from b (over both ways to resolve the corresponding allele of the genotype).

One iteration of this EM algorithm can be done in time $O(nmK^3)$ using the genotype forward algorithm of Equation 3.1 and its backward modification. Even if the derivation of this EM algorithm is a bit tricky, the resulting algorithm is very similar to the basic Baum–Welch algorithm [12, 54].

3.4 Initial Solution

The EM algorithm is only guaranteed to find a locally maximal likelihood solution. Therefore, the initial solution from which the EM is started plays a critical role in the quality of an obtained solution.

In HIT, the transition probabilities in the initial solution are set without looking at the data. First, the transitions from the begin state to the K states at marker 1 are set to $\frac{1}{K}$. Assuming the states at adjacent markers j and $j+1$ are $s_1, \ldots, s_K$ and $t_1, \ldots, t_K$, the transition between states s_l and $t_{l'}$ is set to $\tau(s_l, t_{l'}) = \begin{cases} 1-\rho & \text{, if } l = l' \\ \frac{\rho}{K-1} & \text{, otherwise} \end{cases}$.

This transition setting reflects the idea of founders [66], which was used in Paper I. The model of HIT has K *founder chains* consisting of lth states at every marker (all transition probabilities between adjacent states of this chain are $1-\rho$). These chains model the founder sequences with errors and point–mutations, and the number of transitions between founders are minimized when parameter $\rho < 1 - \frac{1}{K}$, as then the transition probability between the states of a same founder chain is higher than between states of different chains. The value of ρ was 0.1 in the experiments of Paper II and [55].

The algorithm to find the emission parameters is very similar to the one used in the combinatorial method of Paper I. First, the emissions are

discretized by setting them to $1 - \nu$ for the major allele ($\in \{0, 1\}$) and to ν for the other allele. Then these major alleles are assigned greedily from left to right. When column i is decided, each of columns $1, \ldots i-1$ are kept fixed and the alleles at column i maximizing the likelihood of the data up to column i are chosen. After the first greedy round, another round from left to right is made but now both sides of the decided column are taken into account to the likelihood. The time complexity of this algorithm is $O(mnK^2 2^K)$ [55]. The value of ν was 0.01 in the experiments of Paper II and [55].

3.5 Reading the Haplotypes from an HMM

The problem of finding the most probable haplotypes out of the HMM used in HIT is an NP-hard problem [31]. This is not a surprise, as finding the most probable emission sequence (consensus sequence) of an HMM is also NP-hard [42]. In HIT, the haplotypes are read from the model by using a modification of the *Viterbi algorithm* [12, 54]. In this algorithm, two paths through the model are found for each genotype g that together maximize the probability of producing g. Then the order of alleles at each heterozygous position of g can be determined by choosing the order with the highest probability on the Viterbi paths. In [55] this variant of the Viterbi algorithm is improved by sampling paths p and p' for each genotype g from the distribution of $P(p, p'|g, M)$.

3.6 HIT in a Nutshell

The haplotype inference method HIT [55] is the following for given genotypes G and the parameters K and k.

- Find the initial solution with K founders (chains).
- Run the EM algorithm for k iterations.
- Read the haplotypes from the learned HMM and output them.

The parameter k can also be given implicitly as the minimum amount the likelihood must increase in order to continue EM iterations. The total time complexity of HIT is $O(mn)$, if k and K are assumed to be fixed constants. Thus, HIT scales well even to larger data. In Paper II, several restarts were made by making small perturbations to the initial solution (see Figure 2 of Paper II). However, due to the robust behaviour, the current version

of HIT makes only a single start from the initial solution without any perturbations.

3.7 Experimental Results

In the extended version of Paper II [55], the effect of parameter K on the accuracy of the inferred haplotypes was studied. Table 3.1 shows the switch distances obtained with $K = 5, \ldots, 11$ on different datasets.

The datasets CEU-200, YRI-200, CEU-1000, and YRI-1000 were from HapMap database [65] with 120 haplotypes spanning 200 and 1000 SNPs from populations CEU (Utah) and YRI (Yoruba). Pop1 (32 haplotypes), Pop2 (108 haplotypes), and Pop3 (108 haplotypes) were samples from three populations from Finland over 68 SNPs [50, 36]. Also the famous dataset of Daly et al. [9] was included containing 103 SNPs and 258 haplotypes. The known haplotypes of these datasets were converted into genotypes to assess the error (switch distance) in haplotype inference of HIT. Each HapMap dataset (like CEU-200) was actually a set of 22 different datasets and the presented results are averages over these 22 datasets.

It seems that a larger K yields more accurate results at least up to some point. The method HIT* in Table 3.1 selects K by an automatic procedure independently for each genotype. This procedure is explained in more detail in the next subsection.

More experiments with HIT are reported in the next chapter. There the parameter value $K = 10$ was used, as it seems to be a reasonable trade–off between the time spent and the accuracy obtained.

3.7.1 Finding optimal K

Choosing a good number of founders K is a puzzling problem. The common criteria for model selection like AIC, MDL, and BIC favor consistently too small K [55, 58]. Moreover, it seems that overfitting is not a serious problem in haplotype inference [55]. Our solution for automatically selecting K in [55] is partly based on ideas in [58].

This solution uses cross–validation to select the best parameter K independently for each genotype. First, it creates 20 incomplete datasets by marking 10% of heterozygous alleles of the input genotypes as missing. Then HIT is run for each of these 20 datasets and for each $K = 5, \ldots, 11$. For every genotype g a single K is chosen denoted as K_g for which the artificial missing values of incomplete g:s were reconstructed most accurately (compared to the complete g). Finally, the original data is run for each K

	HIT5	HIT6	HIT7	HIT8	HIT9	HIT10	HIT11	HIT*
CEU-200	0.078	0.076	0.073	0.072	0.072	**0.071**	**0.071**	0.72
YRI-200	0.13	0.12	0.12	**0.11**	**0.11**	**0.11**	**0.11**	**0.11**
CEU-1000	0.042	0.038	0.037	0.036	**0.035**	**0.035**	**0.035**	**0.035**
YRI-1000	0.076	0.069	0.063	0.060	0.057	**0.055**	**0.055**	0.057
Pop1	0.21	0.23	0.24	**0.19**	0.22	0.21	0.23	0.21
Pop2	0.17	0.17	0.17	0.17	0.17	**0.16**	**0.16**	0.18
Pop3	0.20	0.21	**0.17**	0.20	0.21	0.20	**0.17**	0.19
Daly et al.	0.029	**0.028**	0.034	0.030	0.031	0.034	0.034	0.030

Table 3.1: Switch distances of HIT with $K = 5, \ldots, 11$ founders on various datasets. In HIT*, parameter K is chosen separately for each genotype using cross–validation. The best results are boldfaced. This table is from [55].

and the haplotypes are chosen for genotype g from the solution with K_g founders.

As shown in Table 3.1, the above procedure does not improve the results significantly. Moreover, the algorithm becomes quite slow as it requires 140 runs. Thus, the problem of finding the best K is not fully solved and is left as an open problem. Similar problem is to find the optimal K in the model of HaploParser (previous chapter).

Chapter 4

A Bayesian Haplotype Inference using Context Tree Weighting

In this chapter, a haplotyping method from Paper III called BACH (Bayesian Context–based Haplotyping) is presented. In BACH, the concept of Context Tree Weighting (CTW), originally used for compressing a single binary string [68], is utilized to efficiently compute the weighted sum over all variable order Markov chains to evaluate the posterior probability (goodness) of a haplotype configuration. The *maximum a posteriori* (MAP) principle is used, i.e. the posterior probability of haplotypes is maximized, in order to find accurate haplotypes for a given set of genotypes. First, some basics of variable order Markov chains are covered.

4.1 Variable Order Markov Chains

In a Markov chain of order D, there are more than N^{D+1} parameters, where N is the alphabet size (number of states). If such a chain is used to model dependencies over long regions, a large D is needed and then the number of parameters becomes very large. Typically there is not enough data to learn all these parameters, as the size of data should be proportional to the number of parameters to be learnt.

A *variable order Markov chain* can be used to allow long *contexts*, i.e. regions taken into account in the distribution of the next symbol, with less parameters. The order of such a chain depends on the context. New parameters are introduced when new contexts are found in the data in sufficient amounts, not automatically to have an equal order on each state. Thus, the complexity (number of variables) of a variable order Markov chain depends on the complexity of the data. In this thesis, *context trees* are used

as the context model. Figure 4.1 gives an example of variable order Markov chain for the DNA alphabet. In this example, the next symbol depends on one or two previous symbols.

Each edge of a context tree corresponds to an alphabet symbol in such a way that each leaf of this tree is a context with variable length (the length is one or two in the example of Figure 4.1). Each leaf of this tree has a distribution of the next symbol, conditional on the corresponding context. Each node is either a leaf, or it has a child for each of the alphabet symbols. In this way, if enough previous symbols are known, the context to be used is uniquely determined.

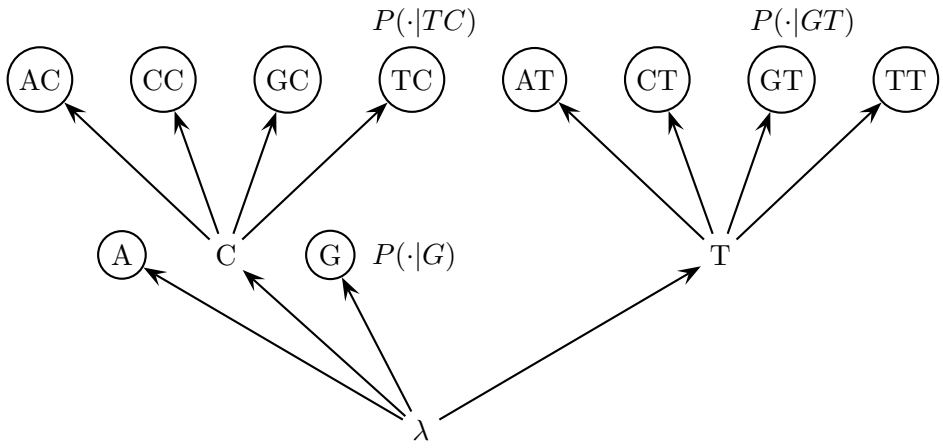

Figure 4.1: A variable order Markov chain as a context tree for the DNA alphabet. Each leaf (circle) has a probability distribution conditional on the context, denoted as $P(\cdot|\text{context})$ (three of which are shown). There are 10 distributions in this model, while in the full second order Markov chain there would be 16 distributions.

4.2 Model of BACH

The Bayesian approach has been used in BACH to model haplotypes and genotypes with variable order Markov chains. The posterior distribution of the haplotypes is composed as

$$P(H|G) \propto P(H,G) = P(G|H)P(H)\,,$$

where the term $P(G|H) = 1$ if the haplotypes H explain genotypes G and $P(G|H) = 0$ otherwise. Thus, the interesting term is $P(H)$ as only haplotypes H explaining genotype G are considered in the haplotype inference problem.

The haplotype model used in BACH is derived from the chain rule that holds for any distribution

$$P(H) = \prod_{j=1}^{m} P(H_j \mid H_1, \ldots, H_{j-1}) \,, \tag{4.1}$$

where H_j is the column (marker) j of the haplotypes H.

Then it is assumed that each H_j depends only on some relatively short context described as a context tree specified by a function c_j (there is a bijection between context trees and functions). The function c_j maps each partial haplotype of $H_1, \ldots, H_{j-1}$ to a particular leaf of the context tree that c_j specifies, i.e. haplotype $h_1, \ldots, h_{j-1} \in H_1, \ldots, H_{j-1}$ is mapped to a haplotype $h_{j-d}, \ldots, h_{j-1}$ for some context length d.

Now the right–hand side of the Equation 4.1 can be written as

$$P(H_j \mid H_1, \ldots, H_{j-1}) = \sum_{c_j} P(c_j) P(H_j \mid c_j(H_1, \ldots, H_{j-1})) \,, \tag{4.2}$$

where $P(c_j)$ is the prior of each function c_j (or equivalently the prior of each context tree). We used the prior $P(c_j) = (1/2)^{N(c_j)}$, where $N(c_j)$ is the number of nodes of depth less than the maximum allowed depth D in the tree c_j.

This prior is the same as in the original CTW method [68]. It is the probability of constructing the binary context tree, by either splitting or stopping at each node independently with probability 1/2. It would be possible to have a more realistic prior, for example, by taking into account the physical distance between the markers in the context.

To complete the model, the term $P(H_j \mid c_j(H_1, \ldots, H_{j-1}))$ needs to be specified. To achieve this, let $S_j(c_j)$ be the set of leafs of the context tree c_j. Attach a parameter θ_s to each leaf $s \in S_j(c_j)$, where s can also be interpreted as a binary string that corresponds to the context. This parameter θ_s gives the probability of allele 1 at marker j with suffix (or context) s. In the original CTW method [68] parameters θ_s are assumed to be independent and having Beta$(1/2, 1/2)$ prior, leading to a closed form

$$P(H_j \mid c_j(H_1 \cdots H_{j-1})) = \prod_{s \in S_j(c_j)} \rho(a_s, b_s) \,, \tag{4.3}$$

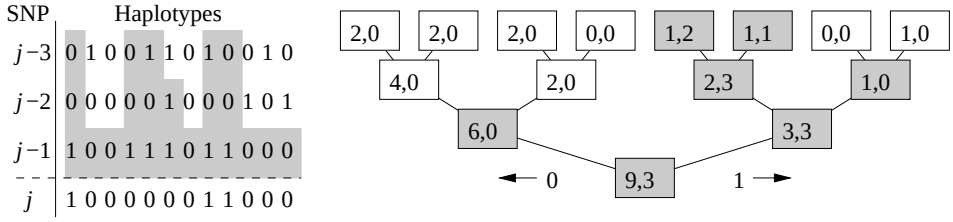

Figure 4.2: The full context tree with maximum depth $D = 3$ for the jth SNP of 12 haplotypes partly shown on the left. The numbers on each node s are the counts a_s and b_s, i.e. the number zeros and ones following the corresponding context. A smaller context tree is shown with gray color. This figure is from Paper III.

where a_s and b_s are the counts of haplotypes up to marker j with suffixes $s0$ and $s1$, respectively. The counts a_s and b_s can be counted from the haplotype data H, as the number of substrings $s0$ and $s1$ ending at the corresponding marker j.

The *leaf score* $\rho(a_s, b_s)$ can be written [67] as

$$\begin{aligned}\rho(a_s, b_s) &= \frac{\Gamma\left(\frac{1}{2} + a_s\right)\Gamma\left(\frac{1}{2} + b_s\right)}{\Gamma\left(\frac{1}{2}\right)\Gamma\left(\frac{1}{2}\right)\Gamma\left(1 + a_s + b_s\right)} \\ &= \frac{\frac{1}{2} \cdot \frac{3}{2} \cdot \frac{5}{2} \cdots (a_s - \frac{1}{2}) \cdot \frac{1}{2} \cdot \frac{3}{2} \cdots (b_s - \frac{1}{2})}{(a_s + b_s)!},\end{aligned} \tag{4.4}$$

where $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$ is the Gamma function. Thus, $\rho(a_s, b_s) \propto \int_0^1 (1 - \theta_s)^{a_s - 1/2} (\theta_s)^{b_s - 1/2} \, d\theta_s$.

However, the experiments of Paper III showed that a simpler prior, namely $\tilde{\rho}(a_s, b_s)$ worked much better in haplotype inference. This prior is defined as

$$\tilde{\rho}(a_s, b_s) = \left(\frac{a_s + \frac{1}{2}}{a_s + b_s + 1}\right)^{a_s} \left(\frac{b_s + \frac{1}{2}}{a_s + b_s + 1}\right)^{b_s}, \tag{4.5}$$

where a_s and b_s are defined as before. Prior $\tilde{\rho}$ can be seen as setting parameter θ_s to the value of $\frac{b_s + \frac{1}{2}}{a_s + b_s + 1}$ (maximum likelihood value with pseudo counts 1/2) instead of integrating it out as is done in Equation 4.4.

4.3 Context Tree Weighting Algorithm

The context trees in BACH are *full binary trees*. In a full binary tree, each node excluding leaves has two children. The number of full binary trees with

maximum depth $\leq D$, denoted by $a(D)$, can be computed from recursion $a(0) = 1$ and $a(D) = a(D-1)^2 + 1$ [1]. Thus, $a(D)$ is of magnitude $2^{2^{D-1}}$ and makes the direct enumeration of Equation 4.2 infeasible for $D > 6$ ($a(7) \approx 4 \cdot 10^{22}$). However, using the context tree weighting algorithm, this sum can be evaluated efficiently for arbitrary large D.

Next, the context tree weighting algorithm is utilized to efficiently compute the sum of Equation 4.2, which is a weighted sum over all context trees. The idea is to compute value ρ_s for each possible haplotype suffix s, as the weighted sum of the leaf score products over all possible subtrees rooted at s. If the length of s is the maximum context depth D, then s must be a leaf and ρ_s is set to $\rho(a_s, b_s)$. Otherwise, ρ_s is obtained by averaging over the case that s is a leaf of a context tree and the case that it has two children, $0s$ and $1s$, i.e.

$$\rho_s := \frac{1}{2}\rho(a_s, b_s) + \frac{1}{2}\rho_{0s}\rho_{1s}\,. \tag{4.6}$$

It can be shown [68] that the dynamic programming recurrence of Equation 4.6 yields

$$\rho_\lambda = \sum_{c_j} 2^{-N(c_j)} \prod_{s \in S_j(c_j)} \rho(a_s, b_s) = P(H_j | H_1, \ldots, H_{j-1})\,, \tag{4.7}$$

where λ is the empty sequence, i.e. the root of each context tree c_j, and $N(c_j)$ is the number of nodes at depth less than D in c_j.

The Equation 4.6 can be evaluated in the (full) tree from which an example is shown in Figure 4.2. It is not necessary to evaluate ρ_s for nodes s that have counts $a_s = b_s = 0$ as $\rho(0,0) = 1$ (and $\tilde{\rho}(0,0) = 1$). Thus, the Equation 4.6 can be evaluated in time $O(nD)$ for a single j, and therefore $P(H)$ can be computed in time $O(mnD)$.

4.4 Simulated Annealing

BACH is based on Bayesian maximum a posteriori (MAP) estimation of haplotypes. Thus, an algorithm is needed to maximize the probability $P(H|G)$, i.e. the probability $P(H)$ of the haplotype configuration H explaining the genotypes G.

A variant of Simulated Annealing (SA), a general heuristic method for global optimization stemming from statistical mechanics [35], has been used to maximize $P(H|G)$.

This method is started from some haplotype configuration H explaining the given genotypes G and then it proceeds iteratively making local changes

(or moves) to the haplotypes of a randomly selected genotype $g \in G$. A single move changes equal–length prefixes of the haplotypes as in the switch distance in Section 1.3.2.

Each proposed move is accepted with probability $\min\{1, A^{1/T}\}$, where A is the ratio of probabilities between the haplotype configuration proposed and the current configuration and T is a temperature parameter. This SA variant is run first from some initial solution with temperature $T = 1$. Then, three more batches with temperatures $T = 1/2, 1/4, 0$ are run and each batch is started from the best solution encountered so far.

The number of iterations made in each batch is mn and each move can be implemented in time $O(D^2)$, thus the total time complexity is (mnD^2).

4.5 Minimization of the Expected Switch Distance

The accuracy of BACH have been improved by combining several haplotype predictions. The switch distance (Section 1.3.2) is used as a loss function to measure the cost of errors in the haplotype predictions. Thus, the final haplotypes are determined by minimizing the expected switch distance to a sample of haplotype configurations, each of which has a locally maximal posterior probability.

The haplotypes minimizing the expected switch distance to a sample of haplotypes are called the *centroid* haplotypes. They can be attained independently for each genotype by a simple voting. To see this, first note that any haplotype pair (h, h') explaining a fixed genotype g can be bijectively mapped to a *switch sequence* $\xi = \xi_1, \xi_2, \ldots, \xi_{k-1}$, where k is the number of heterozygous sites in g and the value of $\xi_i \in \{0, 1\}$ is zero if and only if equal alleles occur in the same haplotype at ith adjacent heterozygous positions. As an example, a genotype 2022 with switch sequences 00, 01, 10, and 11 have the haplotype pairs (0000,1011), (0001,1010), (0011,1000), and (0010,1001), respectively.

If a genotype g has two haplotype configurations with switch sequences ξ and ξ', then the switch distance between the corresponding haplotype pairs is $\sum_{i=1}^{k-1} |\xi_i - \xi'_i| = ||\xi - \xi'||_1$ (1-norm). Thus, the centroid haplotype pair for g can be found as the 1-norm centroid of the binary switch sequences of g in the sample. Such a centroid can be computed by coordinate–wise voting, i.e. the switch sequence of the centroid haplotypes of g has at each position the more common corresponding value in the sample. The sample could be weighted as well (see Paper III).

The centroid haplotype configuration can be found in time linear to the sample size. This problem of finding centroid haplotypes has also been

studied in [58] and in [27].

4.6 Sampling Initial Haplotypes

Simulated annealing is quite sensitive for the initial solution from which it is started. A method called *forward sampling* has been used in Paper III to find a good initial solution from which the SA is started. For a given set of genotypes G and the maximum context length D, it consists of the following steps:

1. Assign haplotypes H^0 randomly to explain genotypes G.
2. Sample haplotype configurations $H^1, \ldots, H^{10}$ by drawing a new haplotype pair (h, h') in H^{i+1} for each genotype $g \in G$ from the distribution $P(h, h'|g, H^i - g)$, where $H^i - g$ is the set H^i without the two haplotypes of genotype g.
3. Return the centroid (Section 4.5) of haplotype configurations $H_5, \ldots, H_{10}$.

The sampling of Step 2 can be implemented with an algorithm similar to forward algorithm of HMMs. This algorithm has a time complexity of $O(mD2^D)$ for a single genotype. As the time complexity depends exponentially on D, it cannot be used for a large D. Parameter D was 8 during the forward sampling in the experiments of Paper III.

4.7 BACH in a Nutshell

The haplotype inference method BACH of Paper III is based on the following iteration for given genotypes G and a parameter D defining the maximum context length.

- Find an initial haplotype configuration with forward sampling (maximum context length = 8).
- Run simulated annealing.
- Output the best haplotype configuration found in the simulated annealing phase.

The above steps are repeated 20 times, with the genotypes being reversed half of the times. Then the final output is a centroid of these 20 runs. The method scales well as the total time complexity depends only linearly on the term mn (with a constant D).

It was noticed during the experiments of Paper III that a higher maximum context length D improved the results. The value of $D = 40$ (in forward sampling $D = 8$) was used as it gave a good accuracy with a reasonable time consumption.

4.8 Experimental Results

In Paper III, 132 real datasets obtained from the HapMap database [65] were used. These datasets contained 120 haplotypes inferred from samples of 30 trios from human populations CEU (Utah) and YRI (Yoruba). From the long haplotypes of each chromosome, 100 SNPs were chosen with an average spacing between adjacent SNPs being 1,3, and 9 kb. Thus, names like CEU-3kb and YRI-9kb refer to datasets from population CEU and YRI with spacing of 3kb and 9kb, respectively.

The number of individuals and haplotypes in the above real datasets were quite modest. We generated 200 datasets with 2000 haplotypes from a 1Mb chromosomal area using COSI software [56]. By filtering, 100 sparse and 100 dense datasets, referred to as Dense and Sparse, were created from the 200 generated datasets. The median SNP counts for Dense and Sparse sets were 367 and 101, respectively.

The known haplotypes of each dataset were converted into genotypes to assess the error in the different haplotype–inference methods including BACH and HIT (Chapter 3). For comparison, PHASE [63, 62, 39], fastPHASE [58], HaploRec [13], and Beagle [4] were also included.

The switch distances with each method are shown in Table 4.1. Datasets Dense and Sparse were too large for PHASE to run in a reasonable time. For the real datasets, PHASE was the most accurate method but also the slowest.

Methods based on HMM (HIT and fastPHASE) do not seem to get the full benefit from the large number of genotypes in datasets Dense and Sparse. However, they work reasonably for smaller HapMap datasets. On the other hand, Beagle based on a single variable order Markov chain does not work well on HapMap datasets but is among the best methods on Dense and Sparse datasets.

BACH seems to work equally well on all datasets. PHASE is a clear winner on real datasets when it comes to accuracy. However, among faster methods BACH is among the most accurate methods on all tested datasets. The runtime of PHASE was several hours on small HapMap datasets. All other methods spent at most few minutes on these datasets. On the biggest simulated datasets the runtime was a few hours.

	PHASE	fastPHASE	BACH	Beagle	HaploRec	HIT
CEU-1k	**0.0299**	*0.0343*	0.0348	0.0405	0.0364	0.0375
CEU-3k	**0.0652**	0.0692	*0.0665*	0.0764	0.0692	0.0745
CEU-9k	**0.144**	*0.146*	0.147	0.164	0.147	0.159
YRI-1k	**0.0407**	0.0579	0.0597	0.0645	*0.0540*	0.0642
YRI-3k	**0.0931**	0.117	0.113	0.125	*0.111*	0.126
YRI-9k	**0.189**	0.204	0.198	0.223	*0.193*	0.220
Sparse	-	0.0398	*0.0305*	0.0317	**0.0288**	0.0442
Dense	-	0.0169	*0.0125*	**0.0116**	0.0133	0.0190
Avg. C+Y	**0.0937**	0.104	0.103	0.116	*0.102*	0.114
Avg. all	-	0.0856	*0.0828*	0.0920	**0.0816**	0.0931

Table 4.1: Switch distances (errors) of the tested methods on various datasets. Average distances over all datasets (Avg. all) and over real datasets (Avg. C+Y = Average over CEU and YRI) are also included. These values are from the experiments of Paper III. The numbers in boldface and in cursive indicate the **best** and the *second best* result, respectively. This table is from Paper III.

Chapter 5

A General Framework for Local Pairwise Alignment Significance with Gaps

In this chapter Paper IV is presented. This paper considers the very basic and fundamental problem of deciding whether local similarities in two sequences occur because the sequences are related or just by chance. The classical statistical hypothesis testing is used, i.e. p-value is used to assess the significance of the best local alignment score. Instead of computing the p-value exactly, a tight upper bound is computed for the p-value. Unlike the previous methods for computing p-values, the presented framework models gaps in the alignments without troublesome sampling and curve fitting, and does not suffer from so–called edge effects (Chapter 1).

5.1 A Novel Dynamic Programming Framework

Here the novel dynamic programming framework from Paper IV is presented. This framework can be used to compute the expected number of alignments of x' and y' distributed according to the (Bernoulli) null model in pseudo–polynomial time. This expectation acts as an upper bound of the p-value.

Let X_t be a random variable counting the number of alignments with score at least t of strings x' and y' distributed according to the null model. The p-value p_t is the probability of picking sequences x' and y' from the null model that have at least one alignment with score $\geq t$, i.e. $p_t = P(X_t \geq 1)$. The expectation $E(X_t)$ can be computed by summing over all alignment paths z and adding up the probability that the score of x' and y' aligned along z is $\geq t$. To justify this, consider a set A_t^z of string pairs (x', y') with

score $\geq t$ aligned along a path z. Now X_t can be written as the sum of indicator functions of A_t^z, i.e. $X_t = \sum_z 1_{A_t^z}$. As the expectation is linear, $E(X_t) = E(\sum_z 1_{A_t^z}) = \sum_z E(1_{A_t^z}) = \sum_z P(A_t^z)$.

The sum of $E(X_t)$ can be computed efficiently as follows. Let $L(i, j, r)$ be the expected number of alignments with score r ending at (i, j) in the alignment grid. Now $L(i, j, r)$ can be computed by dynamic programming by first setting values $L(0, 0, r)$, $L(0, j, r)$, and $L(i, 0, r)$ to one if $r = 0$ and to zero otherwise. The remaining values of $L(\cdot, \cdot, \cdot)$ can be computed from

$$\begin{aligned} L(i, j, 0) &= 1 \\ L(i, j, r) &= L(i-1, j, r+d) + L(i, j-1, r+d) \\ &\quad + \sum_{a,b \in \Sigma} q_a q_b L(i-1, j-1, r - s(a, b)) \,, \end{aligned} \tag{5.1}$$

where $i = 1, \ldots, n$, $j = 1, \ldots, m$, and $r = 1, \ldots, (\min\{m, n\} \max_{a,b} s(a, b))$. Then the expectation $E(X_t)$ can be computed as $E(X_t) = \sum_{i,j} \sum_{r \geq t} L(i, j, r)$. We have obtained an upper bound for the p-value, as by Markov inequality $p_t = P(X_t \geq 1) \leq E(X_t)$.

It is assumed that the values of $L(\cdot, \cdot, \cdot)$ can be presented with sufficient accuracy using a constant–size floating point presentation, and the values of d and $s(\cdot, \cdot)$ are integers with absolute values $\leq B$. Then, evaluating Equation 5.1 needs time $O(nm \min\{m, n\}B)$ and space $O(\min\{m, n\}^2 B)$. Here the term $O(nm \min\{m, n\}B)$ is the number of floating point operations executed, so the assumption of the values being presented with a constant size leads to an algorithm with the same time complexity.

In the bioinformatics domain, it is a feasible assumption that B is a small constant. Moreover, we assume that m and n are of about equal size. Then, the time and space requirements are simply $O(n^3)$ and $O(n^2)$, respectively.

For linear gaps, an algorithm with $O(n^2)$ time and linear space requirement is presented in Paper IV. Equation 5.1 can be generalized to the common *affine gap model*, where each gap has a cost of d for opening the gap and a cost of e for extending it by one symbol. In this case, the time and space requirements are also $O(n^3)$ and $O(n^2)$.

5.2 Experimental Results

In Paper IV, 10^7 random DNA sequence pairs of equal lengths 125, 250, 500, 1000, and 2000 were sampled from the uniform distribution for A, C, G, and T. For each of these sequence pairs, the maximum local alignment

score was computed. The scoring schema was the same that is used in the BLAST, i.e. the score of a match was $+1$, the score of a mismatch -3 [3]. The affine gap model was used, where the gap opening and extension costs were 5 and 2, respectively. The empirical p-values $\hat{p}_t$ were computed from these random sequence pair scores as the fraction of scores that are $\geq t$.

In Figure 5.1 the values of $\hat{p}_t$ and $E(X_t)$ are plotted for sequence lengths 125, 500, and 2000 and for $t = 6, \ldots, 22$. As only 10^7 sequence pairs were sampled, values smaller than 10^{-5} are not accurate. Moreover, the smallest possible $\hat{p}_t$ is 10^{-7}. The ratio of $\hat{p}_t$ and $E(X_t)$ seems to be a constant of about $1/2$, excluding small values of t. In Paper IV, this ratio was suggested to be $(3/4)^2 = c^2$, where $c = \sum_{a,b:s(a,b)<0} q_a q_b$. Thus, the experimental ratio seems to obey what was expected quite well. Reasoning in Paper IV (Section 4.4 of Paper IV) about this ratio could be incorporated into Formula 5.1 leading to an algorithm that computes even a tighter upper bound than $E(X_t)$. This tighter bound will be studied further in the future work.

In Figure 5.2 the values of $E(X_{25})$ are compared to the corresponding p-value computed by the Karlin-Altschul statistics (KA-N). Each result of KA-N is obtained by sampling N random sequence pair scores and then fitting the parameters K and λ to this sample. In this way, the learned K and λ should reduce the edge–effects as well. It seems that the Karlin-Altschul statistics is more accurate for longer sequences and with larger sample size N. The empirical p-values $\hat{p}_{25}$ were computed using the importance sampling method of [46], as the direct sampling of $\hat{p}_{25}$ is not feasible (requires about 10^{13} samples). The available web server implementation allowed only a limited computing time for each user and therefore could not estimate the p-value for sequence lengths of 2000. The error bars in the empirical p-values indicate the standard deviation given by the method. The value of $c^2 E(X_t)$ was included as a close estimate of the real p-value suggested by Figure 5.1. Also this estimate is within the error bars of the empirical p-values. Clearly, $E(X_t)$ behaves more consistently than the Karlin-Altschul statistics.

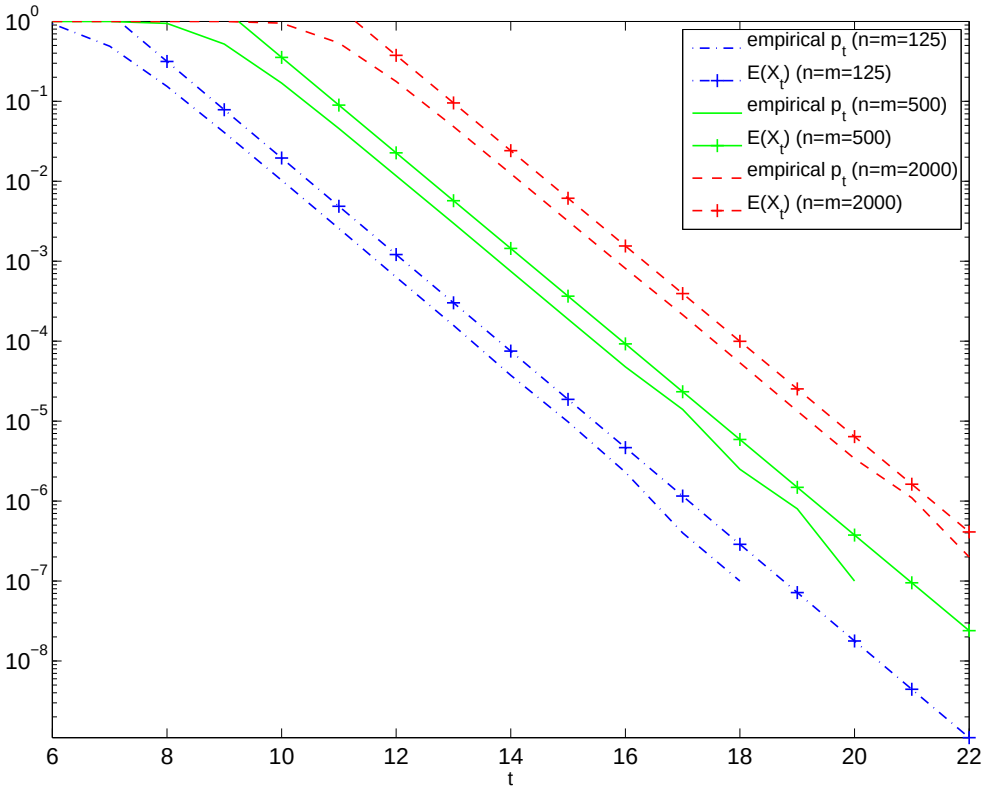

Figure 5.1: The empirical p-value $\hat{p}_t$ and $E(X_t)$ for $t = 6, \ldots, 22$ for sequences with lengths 125, 500, and 2000. The ratio of $\hat{p}_t$ and $E(X_t)$ is very close to $1/2$ expect for the smallest values of t. This figure is based on the experiments of Paper IV.

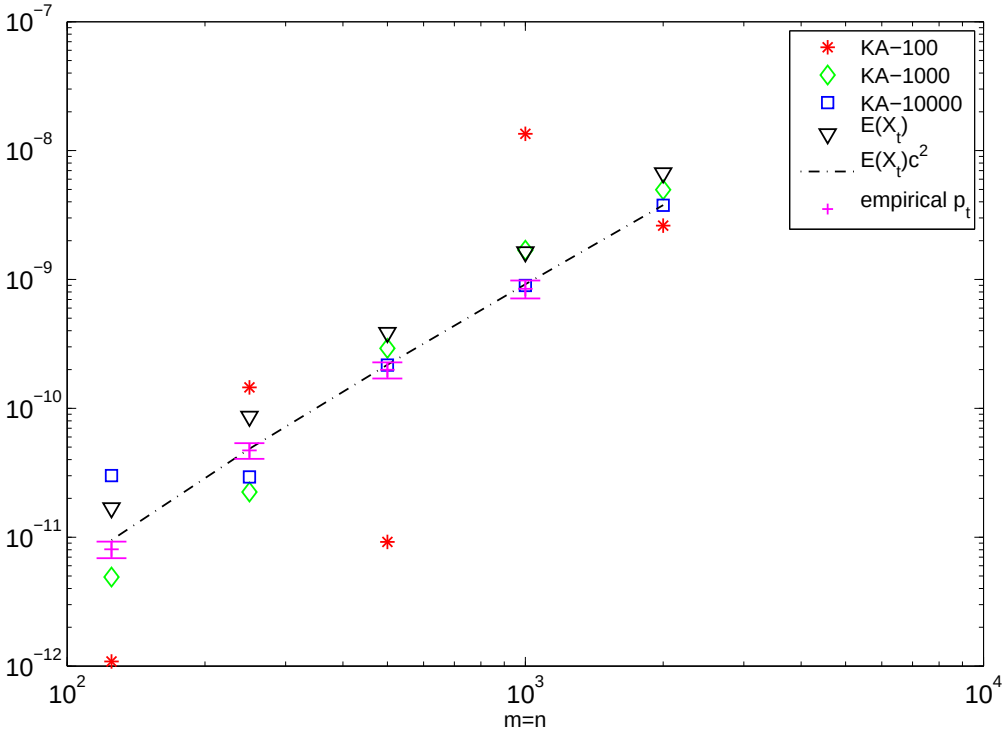

Figure 5.2: Comparison of p-values computed by Karlin-Altschul statistics (KA-N), and the expectation $E(X_t)$ for a score $\geq t = 25$ with varying sequence lengths. The result of KA-N is obtained by fitting the parameters K and λ to a sample of N random sequence pair scores. The value $c^2E(X_t) = \frac{9}{16}E(X_t)$ a good estimate (extrapolated from Figure 5.1) of the real p-value is shown as a dashed line. The empirical p-values (error bars give the standard deviation) are computed by the importance sampling web server of [46] for sequence lengths up to 1000. For sequence lengths of 2000 this was not possible due to the limitations of the server. This figure is partly based on the experiments of Paper IV.

Chapter 6

Conclusions

To choose a method for solving the problem of haplotype inference is not an easy task, as each method has pros and cons. The matters one should consider include, e.g. accuracy, speed, scalability, and the nature of the genotype data.

Typically the speed and accuracy are opposite goals. Most methods could be implemented to trade speed for accuracy, i.e. to run faster but at the expense of the solution quality. Some methods like PHASE are fast for small datasets but do not scale (linearly) to larger datasets. Table 6.1 presents the author's opinion about which methods should be used with respect to the number of SNPs and the number of individuals in the data. Some real datasets might contain large amounts of genotyping errors and missing values, which can affect the performance of the solutions [13]. Moreover, for some datasets the haplotypes are simply easier to infer than for others.

The method of choice presented in this thesis is BACH (Paper III, Chapter 4). It has a good accuracy over different dataset and it scales to large datasets. It has a practical way to cope with missing values (not covered here), and it can model long dependencies even with a small amount of data. However, the speed (and space requirement) of BACH could be improved as the implementation at its current form is not especially fast.

The problem of haplotype inference has been studied a lot both theoretically and in practice. The solutions in Papers I–III use different Markovian models which are very interesting as such. The method of Paper I is not very accurate, but the model itself has its own appeal. In general, probabilistic algorithms seems to yield more accurate results in haplotyping than the combinatorial algorithms. When it comes to speed and accuracy, the implementations of methods used in the results of Papers I–III are not fully optimized. It is probable that these implementations could be improved

with some tedious tweaking. For example, the current version of BACH found from its web page is on average more accurate than the version used in the included experiments.

	m small	m large
n small	PHASE VOMC HMM	VOMC HMM
n large	VOMC Beagle	VOMC Beagle

Table 6.1: A table showing which software should be used for haplotype inference for m SNPs and n individuals. The VOMC includes the methods BACH and HaploRec that are based on variable order Markov chains, and HMM includes fastPHASE and HIT that are based on hidden Markov models. The vertical order of the listed methods gives the preference order of using them (upmost method is the most preferable).

The results with the new alignment score significance framework of Paper IV are promising. The framework is very simple, and still computes tight upper bounds for the p-values. Having an upper bound is an advantage compared to the previous methods that compute an approximate p-value, which can be larger or smaller than the correct one.

The framework is best suited for the linear gap model as then the algorithm is as fast as the Smith–Waterman algorithm having quadratic time with only linear space. Some improvements might be possible to achieve faster algorithms with the commonly used affine gap model.

The query–specific p-values are an interesting problem to study with this approach. In this problem, only one string is randomly distributed according to the null model while the other string is kept fixed.

The Bernoulli null model is a Markov chain of order 0. Thus, a natural direction of future research would be to generalize this framework for higher order null models.

Also the global alignment statistics should be further studied with respect to this framework.

As the high–throughput sequencing has become a reality, the amount of sequence data increases fast [21]. This raises a need for new methods to study these vast amounts of data. The haplotype inference method BACH presented in this thesis is fast and handles large dataset while taking full benefit from large datasets. Thus, it is suitable for high–throughput sequence (genotype) data.

In the local alignment significance computation the increase of sequence data, e.g. number of organisms with the genome sequence in databases, leads to a need of having smaller p-values. The presented novel algorithms for this significance problem are suitable to get accurate estimates and bounds for these small p-values. The speed of these algorithms is sufficient, if the best local alignments scores of interest are computed by a Smith–Waterman type algorithm [59]. Thus, these new algorithms could have applications even in their current (preliminary) form for the analysis of high–throughput sequence data.

All the presented methods are available with their source code, HaploParser via `http://www.cs.helsinki.fi/u/prastas/haploparser`, HIT via `http://www.cs.helsinki.fi/u/prastas/hit`, BACH via `http://www.cs.helsinki.fi/u/prastas/bach`, and the program for alignment score significance via `http://www.cs.helsinki.fi/u/prastas/laswg`.

References

[1] A. Aho and N. Sloane. Some doubly exponential sequences. *Fibonacci Quarterly*, 11:429–437, 1970.

[2] S. Altschul, R. Bundschuh, R. Olsen, and T. Hwa. The estimation of statistical parameters for local alignment score distributions. *Nucleic Acids Research*, 29(2):351–361, January 2001.

[3] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.

[4] S. Browning and B. Browning. Rapid and accurate haplotype phasing and missing-data inference for whole-genome association studies by use of localized haplotype clustering. *American Journal of Human Genetics*, 81(5):1084–97, 2007.

[5] R. Bundschuh. Rapid significance estimation in local sequence alignment with gaps. *Journal of Computational Biology*, 9(2):243–260, 2002.

[6] N. Chia and R. Bundschuh. A practical approach to significance assessment in alignment with gaps. *Journal of Computational Biology*, 13(2):429–441, 2006.

[7] A. Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular Biology and Evolution*, 7:111–122, 1990.

[8] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 1–6, New York, NY, USA, 1987. ACM.

[9] M. Daly, J. Rioux, S. Schaffner, T. Hudson, and E. Lander. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29:229–232, 2001.

[10] M. J. Daly, J. D. Rioux, S. F. Schaffner, T. J. Hudson, and E. S. Lander. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29:229–232, 2001.

[11] Z. Ding, V. Filkov, and D. Gusfield. A linear-time algorithm for the perfect phylogeny haplotyping (pph) problem. In *Research in Computational Molecular Biology (RECOMB)*, pages 585–600, 2005.

[12] R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, first edition, 1998.

[13] L. Eronen, F. Geerts, and H. Toivonen. Haplorec: efficient and accurate large-scale reconstruction of haplotypes. *BMC Bioinformatics*, 7:542, 2006.

[14] L. Excoffier and M. Slatkin. Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology and Evolution*, 12(5):921–927, 1995.

[15] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705 – 708, 1982.

[16] G. Greenspan and D. Geiger. Model-based inference of haplotype block variation. In *Research in Computational Molecular Biology (RECOMB)*, pages 131–137. ACM Press, 2003.

[17] R. Griffiths and P. Marjoram. Ancestral inference from samples of dna sequences with recombination. *Journal of Computational Biology*, 3(4):479–502, 1996.

[18] D. Gusfield. Inference of haplotypes in samples of diploid populations: Complexity and algorithms. *Journal of Computational Biology*, 8(3):305–323, 2001.

[19] D. Gusfield. Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. In *Research in Computational Molecular Biology (RECOMB)*, pages 166–175. ACM Press, 2002.

[20] D. Gusfield. Haplotype inference by pure parsimony. In *Combinatorial Pattern Matching (CPM)*, pages 144–155, 2003.

[21] N. Hall. Advanced sequencing technologies and their wider impact in microbiology. *Journal of Experimental Biology*, 210:1518–1525, 2007.

[22] E. Halperin and E. Eskin. Haplotype reconstruction from genotype data using imperfect phylogeny. *Bioinformatics*, 20(12):104–113, 2004.

[23] J. Hammersley and D. Handscomb. *Monte Carlo Methods*. Fletcher & Son Ltd, Norwich, 1964.

[24] A. Hartmann. Sampling rare events: Statistics of local sequence alignments. *Physical Review E*, 65(5):056102, 2002.

[25] R. Hudson and N. Kaplan. Statistical properties of the number of recombination events in the history of a sample of dna sequences. *Genetics*, 1(111):147–64, 1985.

[26] M. Jensen-Seaman, T. Furey, B. Payseur, Y. Lu, K. Roskin, C.-F. Chen, M. Thomas, D. Haussler, and H. Jacob. Comparative Recombination Rates in the Rat, Mouse, and Human Genomes. *Genome Research*, 14(4):528–538, 2004.

[27] M. Kääriäinen, N. Landwehr, S. Lappalainen, and T. Mielikäinen. Combining haplotypers. Technical Report C-2007-57, Department of Computer Science, University of Helsinki, 2007.

[28] S. Karlin. Statistical signals in bioinformatics. *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 102(38):13355–13362, September 2005.

[29] S. Karlin and S. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 87(6):2264–2268, 1990.

[30] S. Karlin, A. Dembo, and T. Kawabata. Statistical composition of high–scoring segments from molecular sequences. *The Annals of Statistics*, 18(2):571–581, 1990.

[31] J. Kennedy, I. Mandoiu, and B. Pasaniuc. Genotype error detection using hidden markov models of haplotype diversity. In *Algorithms in Bioinformatics (WABI)*, pages 73–84, 2007.

[32] G. Kimmel and R. Shamir. Maximum likelihood resolution of multiblock genotypes. In *Research in Computational Molecular Biology (RECOMB)*, pages 2–9. ACM Press, 2004.

[33] G. Kimmel and R. Shamir. A block-free hidden markov model for genotypes and its application to disease association. *Journal of Computational Biology*, 12(10):1243–1260, 2005.

[34] G. Kimmel and R. Shamir. Gerbil: Genotype resolution and block identification using likelihood. *Proceedings of the National Academy of Sciences of the USA (PNAS)*, 102(1):158–162, 2005.

[35] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[36] M. Koivisto, M. Perola, T. Varilo, W. Hennah, J. Ekelund, M. Lukk, L. Peltonen, E. Ukkonen, and H. Mannila. An mdl method for finding haplotype blocks and for estimating the strength of haplotype block boundaries. In *Pacific Symposium on Biocomputing*, pages 502–513, 2003.

[37] G. Lancia, C. Pinotti, and R. Rizzi. Haplotyping populations: Complexity and approximations. Technical Report DIT-02-0080, Department of Information and Communication Technology, University of Trento, 2002.

[38] N. Landwehr, T. Mielikäinen, L. Eronen, H. Toivonen, and H. Mannila. Constrained hidden markov models for population-based haplotyping. *BMC Bioinformatics*, 8(S-2), 2007.

[39] N. Li and M. Stephens. Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. *Genetics*, 165:2213–2233, 2003.

[40] S. Lin, D. Cutler, M. Zwick, and A. Chakravarti. Haplotype inference in random population samples. *American Journal of Human Genetics*, 71(4):1129–37, 2002.

[41] J. Long, R. Williams, and M. Urbanek. An E-M algorithm and testing strategy for multiple-locus haplotypes. *Americal Journal of Human Genetics*, 56:799–810, 1995.

[42] R. Lyngsø and C. Pedersen. The consensus string problem and the complexity of comparing hidden Markov models. *Journal of Computer and System Sciences*, 65:545–569, 2002.

[43] S. Mercier, D. Cellier, F. Charlot, and J.-J. Daudin. Exact and asymptotic distribution of the local score of one i.i.d. random sequence. In *Proceedings of JOBIM*, pages 74–83, 2000.

[44] Occam's razor n. Merriam-webster's collegiate dictionary (`http://www.merriam-webster.com/dictionary/Occam's\_razor`; accessed 22.09.2009), 2003.

[45] D. Naor and D. Brutlag. On suboptimal alignments of biological sequences. In *Proceedings of CPM*, pages 179–196. Springer-Verlag, 1993.

[46] L. Newberg. Significance of gapped sequence alignments. *Journal of Computational Biology*, 15(9):1187–1194, 2009.

[47] T. Niu, Z. Qin, X. Xu, and J. Liu. Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *American Journal of Human Genetics*, 70(1):157–169, 2002.

[48] K. Palin. *Computational Methods for Locating and Analyzing Conserved Gene Regulatory DNA Elements*. PhD thesis, Department of Computer Science, University of Helsinki, 2007.

[49] B. Pasaniuc, J. Kennedy, and I. Mandoiu. Imputation based local ancestry inference in admixed populations. In *International Symposium on Bioinformatics Research and Applications (ISBRA)*, pages 73–84, 2009.

[50] T. Paunio, J. Ekelund, and T. Varilo *et al.* Genome-wide scan in a nationwide study sample of schizophrenia families in finland reveals susceptibility loci on chromosomes 2q and 5q. *Human Molecular Genetics*, 10:3037–3048, 2001.

[51] W. R. Pearson. Empirical statistical estimates for sequence similarity searches. *Journal of Molecular Biology*, 276(1):71 – 84, 1998.

[52] B. Pierce. *Genetics: A Conceptual Approach*. W. H. Freeman and Company, New York, second edition, 2005.

[53] A. Poleksic, J. Danzer, K. Hambly, and D. Debe. Convergent Island Statistics: a fast method for determining local alignment score significance. *Bioinformatics*, 21(12):2827–2831, 2005.

[54] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[55] P. Rastas, M. Koivisto, H. Mannila, and E. Ukkonen. Phasing genotypes using a hidden markov model. In *Bioinformatics Algorithms: Techniques and Applications*, pages 373–391. Wiley, 2008.

[56] S. Schaffner, C. Foo, S. Gabriel, D. Reich, M. Daly, and D. Altshuler. Calibrating a coalescent simulation of human genome sequence variation. *Genome Research*, 15:1576–1583, 2005.

[57] P. Scheet and M. Stephens. A fast and flexible statistical model for large-scale population genotype data: Applications to inferring missing genotypes and haplotypic phase. *American Journal of Human Genetics*, 78(4):629–644, 2005.

[58] P. Scheet and M. Stephens. A fast and flexible statistical model for large-scale population genotype data: Applications to inferring missing genotypes and haplotypic phase. *American Journal of Human Genetics*, 78(4):629–44, 2006.

[59] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981.

[60] Y. Song and J. Hein. Constructing minimal ancestral recombination graphs. *Journal of Computational Biology*, 12(2):147–169, 2005.

[61] M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9(1):91–116, 1992.

[62] M. Stephens and P. Donnelly. A comparison of bayesian methods for haplotype reconstruction from population genotype data. *American Journal of Human Genetics*, 73(6):1162–1169, 2003.

[63] 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.

[64] T. Strachan and A. Read. *Human Molecular Genetics*. BIOS Scientific Publishers Ltd, Oxford, second edition, 1999.

[65] The International HapMap Consortium. A haplotype map of the human genome. *Nature*, 437:1299–1320, 2005.

[66] E. Ukkonen. Finding founder sequences from a set of recombinants. In *Algorithms in Bioinformatics (WABI)*, pages 277–286. Springer, 2002.

[67] F. Willems. The context-tree weighting method : Extensions. *IEEE Transactions on Information Theory*, 44(2):792–798, 1998.

[68] F. Willems, Y. Shtarkov, and T. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

[69] Y. Wu and D. Gusfield. Improved algorithms for inferring the minimum mosaic of a set of recombinants. In *Combinatorial Pattern Matching (CPM)*, pages 150–161, 2007.