

Finding Consistent Gene Transmission Patterns on Large and Complex Pedigrees

Matti Pirinen and Dario Gasbarra

Department of Mathematics and Statistics,
University of Helsinki, Finland

Final version

November 24, 2005

This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders.

© 20xx IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Abstract

A heuristic algorithm for finding gene transmission patterns on large and complex pedigrees with partially observed genotype data is proposed. The method can be used to generate an initial point for a Markov chain Monte Carlo simulation or to check that the given pedigree and the genotype data are consistent. In small pedigrees the algorithm is exact by exhaustively enumerating all possibilities, but in large pedigrees with considerable amount of unknown data, only a subset of promising configurations can be actually checked. For that purpose the configurations are ordered by combining the approximative conditional probability distribution of the unknown genotypes with the information on the relationships between individuals. We also introduce a way to divide the task in subparts, which has shown to be useful in large pedigrees. The algorithm has been implemented in a program called APE (Allelic Path Explorer) and tested in three different settings with good results.

Index terms: backtracking, heuristic methods, constraint satisfaction, sorting and searching, biology and genetics, pedigree, consistent genotype configuration

1 Introduction

Suppose that a pedigree is given and a part of the individuals' genotypes at a fixed locus is observed. We propose a new method for extending partially known genetic data to the whole pedigree. The main motivation for this work is that Markov chain Monte Carlo (McMC) analyses of genetic data on pedigrees need one consistent configuration to start with. McMC methods are used for the estimation of quantities whose exact calculation is impractical and recently there has been much interest in developing McMC samplers also in genetics [17]. Another application of our algorithm is that by finding a consistent configuration we actually prove the data consistent. The consistency of data is not self-evident in large pedigrees, because there may be genotyping errors or the familial relationships may be incorrectly specified (e.g. false paternity). As the first step of genetic analysis one should be able to determine that the data are at least Mendelian consistent.

In the realm of computer science we are working with an instance of constraint satisfaction problems (CSP) [13], [14]. Namely, the alleles of the individuals in the pedigree can be considered as variables for which the observed data and the Mendelian rules of inheritance pose logical constraints. In general a CSP requires assigning such values to variables that satisfy constraints or establishing that no such assignment exists. However, for the problem at hand, our method can prove the consistency of the data only by finding an admissible configuration and cannot establish its possible inconsistency except for in fairly limited cases in which an exhaustive search is computationally possible. Indeed, Aceto et al.[1] proved that in general consistency checking on pedigrees with marker data containing at least three alleles is an NP-complete problem.

In simple pedigrees that do not contain loops, well-known ideas of genotype elimination [9],[15] and peeling [2],[18] can be modified to solve the problem efficiently. We briefly summarize these ideas as well as the approaches of Heath [5] and Luo and Lin [12] that are specifically designed for finding consistent starting points for McMC samplers.

Genotype elimination is based on the marginal conditional distributions, given the

overall data, of the genotypes of the individuals in the pedigree. The pedigree is compatible with the data if and only if each individual has a nonempty list of genotypes with positive marginal conditional probabilities.

The Lange-Goradia algorithm [9] and its extensions [15] are implemented in the program PedCheck which computes the lists of genotypes with positive marginal conditional probability by using a sum-product message-passing algorithm (see also [10]).

For pedigrees with loops, PedCheck determines first a set of loop-breakers and fixes their genotypes before performing the genotype elimination separately on each subgraph. This is done until genotype elimination is successful for all subgraphs, or the list of joint loop-breaker genotype configurations is exhausted. If the pedigree is not compatible with the data, in the worst case the number of times we have to run the sum-product algorithm, before we obtain a negative answer, grows exponentially with the number of loop-breakers.

Note also that, when genotype elimination is successful, we know only that a jointly compatible genotype configuration exists. It is possible to find one by repeating the following two steps as long as more than one individual has multiple possibilities in his genotype list: i) fix a possible genotype for some individual whose eliminated list still has multiple entries; ii) run the sum-product algorithm again to update the genotype lists. However, it is not clear whether this method is efficient in complex pedigrees. In the Results section we have compared the running times of our algorithm with those of the program PedCheck.

In 1971 Elston and Stewart proposed an algorithm [2] that calculates efficiently genetic likelihoods on loopless pedigrees. Variants of this algorithm are known as 'peeling' methods since they 'peel out' peripheral families from the pedigree by representing the likelihood of the family as a function of the genotype of a pivot individual, who connects the family to the unpeeled part of the pedigree. If a pedigree is successfully peeled, we can sample the genotypes for individuals in reverse peeling order from the exact genotype distributions and thus generate a consistent configuration for the given data [5]. However, peeling is not practical in pedigrees that contain many loops and a lot of unknown data

as is shown in Results, where we have tested some data sets on the preparation step of the Loki program [6],[7] that implements the peeling method for both checking consistency and generating an initial state for McMC analysis.

For complex pedigrees with loops, Heath introduced an algorithm [5] that uses a mixture of peeling and genotype elimination to approximate the distribution of the genetic data on pedigrees. The idea is to peel families whenever possible and otherwise sample genotypes of families from a certain approximating distribution. Due to inexactness, the method is not guaranteed to produce a proper sample. In [5] the fractions of consistent configurations for two large pedigrees (1373-individuals and 1600-individuals) were reported to be 50% and 69%, respectively. Since the program is not publicly available (personal communication with Dr. Heath), we could not run it in other settings.

Luo and Lin [12] have implemented a totally different method in a program called START. It generalizes the heated Gibbs sampler with relaxed penetrance (HGRP), which was introduced by Lin et al. [11] and uses the single-site Gibbs sampler [4] to update the genotypes, one by one, conditioned on the observed phenotypes and the data of neighbouring individuals. Luo and Lin [12] tested START with two Hutterite pedigrees (one with 221 and the other with 1544 individuals). We also used the smaller Hutterite pedigree in testing our program.

The basic idea in our method is that of backtracking [13]: we start from the bottom of the pedigree and extend a partial consistent configuration, one level at a time, until founders are reached. Whenever an inconsistency is encountered, the algorithm changes the partial configuration at some lower level. As the number of possibilities in large pedigrees with considerable amount of unknown data is astronomical, the key idea in our method is an ordering criterion by which we choose the most promising configurations for actual checking. In large pedigrees it has also shown to be helpful to be able to concentrate on smaller parts of the pedigree at a time, and use the partial results as additional data in further analyses.

We have implemented our method in a program called APE (Allelic Path Explorer).

The three test cases explained in Results consider both real (221-individuals) and simulated (from 1500 to 4921 individuals) pedigrees with simulated partial data. Comparison with the results reported by Luo and Lin [12] suggests that the performance of our method, measured in running time, is less affected by the increase in the number of alleles. Absolute comparison between the two is difficult because the performances of both programs depend on (many) parameter values.

2 Definitions

By a pedigree we mean a directed graph that specifies the family relationships between individuals. We draw pedigrees as marriage node graphs (e.g. Figure 1) in which there are two kinds of nodes: individual nodes (squares for males and circles for females) and mating nodes (small black circles). Each mating node represents a nuclear family connecting two parents to their common offspring. The directions of edges correspond to the flow of genetic material i.e. from parents to mating nodes and from mating nodes to offspring. In our figures the directions of the edges are determined by the fact that parents are depicted above the corresponding mating node whereas offspring lie below it. A more formal definition of pedigrees can be found for example in [1].

We assume that a pedigree is given and denote by \mathcal{J} the set of individuals in it. The individuals whose parents are not included in the pedigree are called *founders*. By convention every individual i , who is not a founder, has both of his/her parents in the pedigree and we denote by f_i and m_i the father and the mother of i , respectively.

We also assume that some (partial) genotype information of the individuals is given. As we are interested only in generating one Mendelian consistent genotype configuration, we consider a single locus at a time, whence the Mendelian rules of inheritance state that each individual carries two alleles and each nonfounder inherits exactly one allele from his father and the other one from his mother. We assume that there exist more than two alleles of the gene among the known data. This is because for biallelic data we can always

create a consistent configuration by taking those individuals as heterozygotes who are not forced to be homozygotes by the known data [5]. We denote by \mathcal{A} the set of labels for different alleles. We also include the symbol \emptyset in \mathcal{A} to mark the alleles whose type is still unknown to us and say that these alleles are *undefined*. Alleles with labels different from \emptyset are called *defined*.

We denote the set of unordered partial genotypes by \mathcal{G} . Thus

$$\mathcal{G} = \{\{a, b\} : a, b \in \mathcal{A}\}.$$

2.1 Configurations and descent paths

By a *genotype configuration* on the pedigree we mean a mapping $\tau : \mathcal{J} \rightarrow \mathcal{G}$. Observed genotypes are unordered, but for notational reasons we may impose an arbitrary ordering on each genotype and consider the corresponding genotype configuration as a mapping from \mathcal{J} to \mathcal{A}^2 .

Let τ be an ordered genotype configuration, $i \in \mathcal{J}$ and $r \in \{1, 2\}$. A *descent path* of the allele $\tau_r(i)$ is a sequence $(i_0, i_1, \dots, i_k) \in \mathcal{J}^{k+1}$, where $k \geq 0$, $i_0 = i$, i_k is a founder and $i_{h+1} \in \{f_{i_h}, m_{i_h}\}$ for each $h < k$. Thus a descent path tracks the origins of a particular allele up to the founders. Note that for each nonfounder there are several possible descent paths, since at each step a path can follow either the paternal or the maternal line of descent.

We denote by τ_{data} the genotype configuration that corresponds to the observed data. We are interested in finding a consistent set of descent paths for the defined alleles of τ_{data} . More precisely, consistency requires that for each individual one allele is inherited from the father and the other one from the mother, and that every individual is a member of at most two descent paths of different kinds of alleles.

2.2 Levels and components

The appropriate step size in our algorithm is that of a level. In pedigrees with non-overlapping generations levels are simply generations. However, we need a more general concept in complex pedigrees where generations are not well-defined.

Define recursively the *rank* ρ of an individual i as

$$\rho(i) = \begin{cases} 0, & \text{if } i \text{ is a founder.} \\ \max\{\rho(f_i), \rho(m_i)\} + 1, & \text{otherwise.} \end{cases}$$

The rank is the largest number of mating nodes encountered on a path from the individual to any of his ancestors in the pedigree. Let $L = \max\{\rho(i) : i \in \mathcal{J}\}$ and for each $i \in \mathcal{J}$, define the *level* as $l(i) = L - \rho(i)$. Thus for the individuals at level ℓ , a maximal ancestral path is of length $L - \ell$. In particular, the highest level L consists of the founders. We assume that $L > 0$ in the given pedigree.

The search for descent paths proceeds levelwise, and each level in turn is divided into independent components. Let $\ell < L$. Consider the individuals at level ℓ together with their parents, i.e. the sets

$$C_\ell = \{i \in \mathcal{J} : l(i) = \ell\}$$

and

$$P_\ell = \{i \in \mathcal{J} : i = f_j \text{ or } i = m_j \text{ for some } j \in C_\ell\}.$$

Note that C_k and P_ℓ are disjoint for all $k \leq \ell$, since by definition $l(i) > \ell$ for all $i \in P_\ell$. Note also that every nonfounder belongs to exactly one of the sets C_k but an individual may belong to several of the sets P_k where $k < L$. We can define an undirected bipartite graph on the set $C_\ell \cup P_\ell$ in which there is an edge between $c \in C_\ell$ and $p \in P_\ell$, if and only if $p \in \{f_c, m_c\}$. Let \mathcal{K}_ℓ be the set of connected components of this bipartite graph. We can consider each component independently when we extend the paths of the defined alleles from the individuals in C_ℓ to their parents in P_ℓ .

3 Searching for consistent descent paths

To search for a consistent set of descent paths, we proceed level by level from 0 up to L . If at some level $\ell < L$ the temporary configuration we are working with cannot be extended to the set P_ℓ , we have to change it at some lower level(s). We describe here a backtracking algorithm that does this in a systematic way. The concise formulation is given in Appendix A. To make things computationally feasible also in large pedigrees, section 3.2.3 introduces some heuristics for the actual implementation.

3.1 Travelling the pedigree

By a configuration for the set P_ℓ we mean a mapping $\sigma_\ell : P_\ell \rightarrow \mathcal{G}$, whereas a temporary configuration $\bar{\sigma} = (\sigma_0, \dots, \sigma_{\ell-1})$ at level ℓ is a genotype configuration which is compatible both with the observed data and the configurations $\sigma_k : P_k \rightarrow \mathcal{G}$ for each $k < \ell$, and which leaves the remaining genotypes on the pedigree undefined.

To be more precise, we recursively define the sets

$$S_{(\sigma_0, \sigma_1, \dots, \sigma_{\ell-1})}^\ell \subseteq \{\sigma : P_\ell \rightarrow \mathcal{G}\},$$

where $\ell < L$ and each $\sigma_k \in S_{(\sigma_0, \sigma_1, \dots, \sigma_{k-1})}^k$, for $k < \ell$.

Let S^0 contain those configurations for the set P_0 that have all the observed alleles of the individuals in C_0 transmitted to their parents in P_0 and that are consistent with the laws of inheritance and the observed data.

Suppose $\ell < L$ and that $(\sigma_0, \dots, \sigma_{\ell-1})$ is such a temporary configuration that $\sigma_k \in S_{(\sigma_0, \dots, \sigma_{k-1})}^k$ for all $k < \ell$. Note that the individuals in C_ℓ can have offspring only in some C_k where $k < \ell$, hence every defined allele from the lower levels has already found its way at least to level ℓ . Define $S_{(\sigma_0, \sigma_1, \dots, \sigma_{\ell-1})}^\ell$ as the set of consistent configurations for P_ℓ conditioned on the fact that the genotypes of the individuals in P_k are consistent with the mapping σ_k for all $k < \ell$, and that all defined alleles of the individuals in C_ℓ have been transmitted to the corresponding parents in P_ℓ .

The goal of our algorithm is to construct a temporary configuration $(\sigma_0, \dots, \sigma_{L-1})$, where $\sigma_k \in S_{(\sigma_0, \dots, \sigma_{k-1})}^k$ for each $k < L$. Such a configuration transmits all the defined alleles up to the founders in a consistent way and thus contains enough information for finding a consistent inheritance pattern on the pedigree. The construction of a suitable temporary configuration proceeds levelwise from 0 to L .

Suppose we have reached level $\ell < L$ with the temporary configuration $(\sigma_0, \dots, \sigma_{\ell-1})$ and stored the sets $S_{(\sigma_0, \dots, \sigma_{k-1})}^k$ with $k < \ell$. Our next task is to find the set $S_{(\sigma_0, \dots, \sigma_{\ell-1})}^\ell$ and decide how to proceed depending on whether this set is empty.

If $S_{(\sigma_0, \dots, \sigma_{\ell-1})}^\ell$ is empty, then the temporary configuration $(\sigma_0, \dots, \sigma_{\ell-1})$ cannot be extended to the set P_ℓ and must be modified. We choose the largest positive $k < \ell$ for which there exists an unchecked configuration in the set $S_{(\sigma_0, \dots, \sigma_{k-1})}^k$, pick the first unchecked configuration σ'_k and continue with the temporary configuration $(\sigma_0, \dots, \sigma_{k-1}, \sigma'_k)$ at level $k + 1$. If no such k exists, then the data are inconsistent with the pedigree.

On the other hand, if $S_{(\sigma_0, \dots, \sigma_{\ell-1})}^\ell$ is not empty, we order it according to the rules that will be explained later and choose the first configuration from it to extend the temporary configuration to level $\ell + 1$.

Let us take a closer look at how to find the set $S_{(\sigma_0, \dots, \sigma_{\ell-1})}^\ell$. First we check the known data for P_ℓ . Data may be known for two reasons: the parent's genotype may be observed or the parent may have a child also at some lower level $k < \ell$ and thus the mapping σ_k may already have transmitted some alleles to him.

Then we proceed componentwise to find all consistent configurations. If a component has n children, it can have at most 2^n parental configurations. Usually however, there are much fewer configurations due to the observed data of parents, undefined data of offspring, zero conditional allelic frequencies (defined in (3.2.1)), and the fact that a parent cannot carry more than two different alleles. Furthermore, some genotype configurations are redundant, whence we shall save only those with the minimum number of defined alleles.

We are interested in the (reverse) transmission of the observed alleles up to the founders and leave those genotypes undefined that are not part of these descent paths. However,

after the descent paths of the known alleles have been found, we may fill the pedigree by sampling the undefined genes for founders and dropping them down through the pedigree according to some consistent inheritance pattern.

3.2 Ordering the configurations

Once we have found all consistent configurations for P_ℓ compatible with the temporary data at the lower levels, we define an ordering among the configurations. We use two different measures in this process, namely the pseudo-conditional genotype probabilities and the amount of genetic variation between relatives.

3.2.1 Pseudo-conditional genotype probabilities

We assume that the considered population is in Hardy-Weinberg equilibrium with known allelic frequencies, denoted by $fr(a)$ for $a \in \mathcal{A} \setminus \{\emptyset\}$. Starting from the founder level, we recursively define the pseudo-conditional genotype probability $\pi_i(\{a, b\})$ as an approximation to the conditional probability that the individual i has the genotype $\{a, b\}$ given the data available on him and his ancestors. This is an approximation since it is taken as if the genes were transmitted independently from parents to their children, which is the case only when the genotypes of the parents are completely known, or the parents are unrelated with respect to the pedigree.

In the following definition $\{g_1, g_2\}$ refers to the (partially) observed genotype of the individual i .

If i is a founder and $a, b \neq \emptyset$, we set

$$\begin{aligned} \pi_i(\{a, b\}) &= (1_{(g_1=a)} + 1_{(g_1=\emptyset)}fr(a)) \times (1_{(g_2=b)} + 1_{(g_2=\emptyset)}fr(b)) \\ &+ 1_{(b \neq a)} \times (1_{(g_1=b)} + 1_{(g_1=\emptyset)}fr(b)) \times (1_{(g_2=a)} + 1_{(g_2=\emptyset)}fr(a)). \end{aligned}$$

For a generic individual at some lower level, we compute first an additional $\tilde{\pi}_i$ -distribution

by setting for $a, b \neq \emptyset$

$$\begin{aligned} \tilde{\pi}_i(\{a, b\}) &= Pr(f_i \text{ transmits } a) \times Pr(m_i \text{ transmits } b) \\ &+ 1_{(a \neq b)} \times Pr(f_i \text{ transmits } b) \times Pr(m_i \text{ transmits } a), \text{ where} \end{aligned}$$

$$Pr(j \text{ transmits } a) = \frac{1}{2} \left[\sum_{c \neq \emptyset} \pi_j(\{a, c\}) + \pi_j(\{a, a\}) \right]. \quad (3.1)$$

Here $Pr(f_i \text{ transmits } a, m_i \text{ transmits } b)$ is replaced by the product $Pr(f_i \text{ transmits } a) \times Pr(m_i \text{ transmits } b)$. Without this approximation, we would need to compute the conditional genotype probabilities jointly for all individuals at the same level. Note that $\tilde{\pi}_i(\{a, b\}) = 0$ implies that i cannot have the genotype $\{a, b\}$ given the pedigree and the data on his ancestors. Thus $\tilde{\pi}_i$ can be used to find some (but not all) inconsistencies between the observed data and the pedigree.

Now we obtain π_i by conditioning $\tilde{\pi}_i$ on the partially observed genotype $\{g_1, g_2\}$ as follows.

If $\{g_1, g_2\} = \{\emptyset, \emptyset\}$, we just have $\pi_i = \tilde{\pi}_i$.

If $g_1, g_2 \neq \emptyset$ and $\tilde{\pi}_i(\{g_1, g_2\}) > 0$, we set $\pi_i = \delta_{\{g_1, g_2\}}$.¹

If $g_1, g_2 \neq \emptyset$ and $\tilde{\pi}_i(\{g_1, g_2\}) = 0$, we have found a contradiction: the observed genotypes are not compatible with the pedigree.

If $g_1 \neq \emptyset$ and $g_2 = \emptyset$, we have to condition on the event $E_i(g_1) = \{\text{"a randomly chosen allele from the individual } i \text{ turns out to be } g_1\}$. First we note that if $\sum_{c \neq \emptyset} \tilde{\pi}_i(\{g_1, c\}) = 0$,

$${}^1\delta_{\{g_1, g_2\}}(\{a, b\}) = \begin{cases} 1, & \text{if } \{a, b\} = \{g_1, g_2\}, \\ 0, & \text{otherwise.} \end{cases}$$

we have found a contradiction between the pedigree and the data. Otherwise we set

$$\begin{aligned}
\pi_i(\{a, b\}) &= Pr(\{a, b\} | E_i(g_1)) \\
&= \frac{Pr(E_i(g_1) | \{a, b\}) Pr(\{a, b\})}{Pr(E_i(g_1))} \\
&= \frac{1_{(g_1 \in \{a, b\})} (\frac{1}{2} + \frac{1}{2} 1_{(a=b)}) \tilde{\pi}_i(\{a, b\})}{\frac{1}{2} \tilde{S}_i(g_1)} \\
&= \begin{cases} \frac{2\tilde{\pi}_i(\{g_1, g_1\})}{\tilde{S}_i(g_1)}, & \text{if } a = b = g_1, \\ \frac{\tilde{\pi}_i(\{a, b\})}{\tilde{S}_i(g_1)}, & \text{if } a \neq b \text{ and } g_1 \in \{a, b\}, \\ 0, & \text{otherwise,} \end{cases}
\end{aligned}$$

where $\tilde{S}_i(g_1) = \sum_{c \neq \emptyset} \tilde{\pi}_i(\{g_1, c\}) + \tilde{\pi}_i(\{g_1, g_1\})$.

The case $g_2 \neq \emptyset, g_1 = \emptyset$ is treated accordingly.

We extend the definition of π_i also to the partially observed genotypes by defining $\pi_i(\{\emptyset, \emptyset\}) = 1$ and $\pi_i(\{a, \emptyset\}) = Pr(E_i(a))$ for all $a \neq \emptyset$. Note that $Pr(E_i(a))$ equals $Pr(i \text{ transmits } a)$ calculated in (3.1).

Now we may define the probability model for configurations at a single level. Since the components in \mathcal{K}_ℓ are independent with respect to the transmission of alleles, first we define the probabilities for configurations in one component and then extend the model to the whole set of parents by taking products.

Let $K \in \mathcal{K}_\ell$ and let $P_K \subseteq P_\ell$ be the set of parents in the component K . Let $S_K \subseteq \{s : P_K \rightarrow \mathcal{G}\}$ be the set of consistent configurations on P_K . We define the pseudo-probability of $s \in S_K$ as

$$p_K(s) = \frac{\prod_{i \in P_K} \pi_i(s(i))}{\sum_{s' \in S_K} \prod_{i \in P_K} \pi_i(s'(i))}. \quad (3.2)$$

For each configuration $\sigma \in S_\sigma^\ell$, we set its probability to be

$$p(\sigma) = \prod_{K \in \mathcal{K}_\ell} p_K(\sigma|_{P_K}), \quad (3.3)$$

where $\sigma|_{P_K}$ is the restriction of σ on P_K .

3.2.2 Variation coefficients

By introducing the variation coefficients we try to minimize the amount of genetic variation between close relatives. This is important especially when there is little observed data at higher levels of the pedigree.

Let us recall that the kinship coefficient Φ_{ij} is defined as the prior probability that a randomly chosen allele from i and a randomly chosen allele from the same autosomal locus of j are identical by descent. By prior probability it is meant that only the pedigree counts, not the observed genotypes. The algorithm for computing Φ_{ij} can be found in [8]. Here we use these coefficients as a measure of the relatedness of individuals.

We also have to quantify the difference d between two genotypes taking into account possibly undefined alleles. Let σ be a genotype configuration whose domain includes individuals i and j . We denote by $h(i, j) \in \{0, \dots, 4\}$ the number of those defined alleles that only one of i and j carries under σ . Note that if an individual is a homozygote, both of his alleles contribute to the value of h separately. We mark by $u(i, j) \in \{0, \dots, 4\}$ the number of undefined alleles among the genotypes $\sigma(i)$ and $\sigma(j)$. It follows that $4 - u(i, j)$ alleles among $\sigma(i)$ and $\sigma(j)$ are defined. If one of i and j has a totally undefined genotype or if the genotypes coincide, we set $d_\sigma(i, j) = 0$. Otherwise we define

$$d_\sigma(i, j) = h(i, j) + (4 - u(i, j)) - u(i, j) = 4 + h(i, j) - 2u(i, j).$$

The nonzero values of d_σ and the corresponding genotypes can be found in Table I. The relevance of this definition can be seen by considering i and j as children of the same nuclear family: the larger the difference $d_\sigma(i, j)$, the more strictly i and j together determine the genotypes of their parents.

In general, by choosing configurations that reduce the genetic variation among siblings, we are more likely to find consistent configurations for the corresponding parents. With this in mind, we define the variation coefficient $v(I, \sigma)$ for any set I of individuals and

Table I: Nonzero values of d .

$\sigma(i)$	$\sigma(j)$	$d_\sigma(i, j)$
$\{a, \emptyset\}$	$\{b, \emptyset\}$	2
$\{a, a\}$	$\{a, \emptyset\}$	2
$\{a, b\}$	$\{a, \emptyset\}$	3
$\{a, a\}$	$\{a, b\}$	5
$\{a, a\}$	$\{b, \emptyset\}$	5
$\{a, b\}$	$\{c, \emptyset\}$	5
$\{a, b\}$	$\{a, c\}$	6
$\{a, a\}$	$\{b, b\}$	8
$\{a, a\}$	$\{b, c\}$	8
$\{a, b\}$	$\{c, d\}$	8

any partially observed genotype configuration σ whose domain includes I as

$$v(I, \sigma) = \sum_{i \in I} \sum_{j \in I \setminus \{i\}} \Phi_{ij} d_\sigma(i, j)^t,$$

where t is a fixed parameter.

In our final ordering criterion we combine the pseudo-probabilities and the variation coefficients. We find the descending ordering for the configurations $\sigma_\ell \in S_\sigma^\ell$ according to the scores

$$p(\sigma_\ell) \exp(-\kappa v(P_\ell, \sigma_\ell)) \tag{3.4}$$

where $\kappa \in [0, \infty[$ is a fixed parameter that determines the weight of the variation coefficients in the ordering process.

3.2.3 Ordering in practice

The set of possible configurations can easily be found for each component separately, but when considering the whole level, product must be taken over all components, which may

result in so many combinations that exact ordering according to (3.4) is impossible in practice. Therefore we proceed by merging components one by one into a cluster and possibly dropping some configurations out of further considerations every time a new component has been added to the cluster.

Suppose we are about to order the set S_{σ}^{ℓ} . Note that (3.4) can be factored as

$$\prod_{k \leq n} \left[p_{K_k}(\sigma_{\ell|P_{K_k}}) \text{Int}(K_k; \sigma_{\ell}) \prod_{m < k} \text{Ext}(K_k, K_m; \sigma_{\ell}) \right] \quad (3.5)$$

where components of the level ℓ are enumerated as K_1, \dots, K_n , the term $\text{Int}(K_k; \sigma_{\ell}) = \exp(-\kappa v(P_{K_k}, \sigma_{\ell}))$ contains the interactions within the component K_k and the term $\text{Ext}(K_k, K_m; \sigma_{\ell}) = \exp(-\kappa \sum_{i \in P_{K_k}} \sum_{j \in P_{K_m}} \Phi_{ij} d_{\sigma_{\ell}}(i, j)^t)$ contains the interactions between the components K_k and K_m .

The enumeration we use for the merging process is such that for all $m \leq m' \leq n$ the sum of the kinship coefficients between different parents in the set $\bigcup_{k < m} P_{K_k} \cup P_{K_{m'}}$ is maximal when $m' = m$. We also introduce three upper bounds M_{clu} , M_{comp} and M_{lev} for cluster, component and level, respectively.

The first step in the merging process is to find the decreasing order among the configurations σ defined on the component K_1 according to the temporary scores

$$p_{K_1}(\sigma_{|P_{K_1}}) \text{Int}(K_1; \sigma). \quad (3.6)$$

We shall keep in memory at most M_{clu} of these and say that at this point the cluster consists of the component K_1 .

Suppose we have merged components K_1, \dots, K_m , where $m < n$, into the cluster and stored $c \leq M_{clu}$ product-configurations for the corresponding set of parents $\bigcup_{k \leq m} P_{K_k}$. For each configuration σ we also have computed the temporary score

$$\prod_{i \leq m} \left[p_{K_i}(\sigma_{|P_{K_i}}) \text{Int}(K_i; \sigma) \prod_{j < i} \text{Ext}(K_i, K_j; \sigma) \right]. \quad (3.7)$$

Our next task is to merge the component K_{m+1} into the cluster. We assume that there are $c' \leq M_{comp}$ configurations in the component K_{m+1} , (otherwise we shall consider only

the first M_{comp} of them, chosen with a similar formula to that introduced in (3.6)). We shall find the descending order for the $c \times c' \leq M_{clu} \times M_{comp}$ product-configurations σ defined on the set $P = \bigcup_{k \leq m+1} P_{K_k}$ according to the scores

$$\exp(-\kappa v(P, \sigma)) \prod_{k \leq m+1} p_{K_k}(\sigma|_{P_{K_k}}). \quad (3.8)$$

Note that in order to calculate (3.8) we only have to multiply the existing product-configuration scores (3.7) with the corresponding expressions

$$p_{K_{m+1}}(\sigma|_{P_{K_{m+1}}}) Int(K_{m+1}; \sigma) \prod_{i \leq m} Ext(K_{m+1}, K_i; \sigma).$$

Now the cluster contains the components K_1, \dots, K_{m+1} and if $c \times c' > M_{clu}$, we shall reduce the number of configurations in the cluster to M_{clu} .

After all components have been merged into the cluster, we may still reduce the number of product-configurations so that it is at most M_{lev} . This makes rapid moving between different levels possible.

Note that if no restrictions are made ($M_{clu} = M_{comp} = M_{lev} = \infty$) the algorithm does an exhaustive search. However, this is possible only in very limited cases of small pedigrees or almost totally observed data.

3.3 Extracting components

If the algorithm does not find a consistent configuration, it outputs the components in which most of the inconsistencies were encountered and we may concentrate solely on those parts of the pedigree. We refer to this procedure as the extraction of components.

For each component K let us define its offspring $O(K)$ as the set of individuals who are descendants of some individual in K . Thus every observed allele whose descent path may go through the component K belongs to the known data of some individual in $O(K)$.

When extracting component K we simply set unknown all genotypes of the individuals outside $O(K)$. This applies to the observed data as well as to the alleles that are transmitted upwards in the pedigree as the algorithm proceeds. The idea is that we do not keep

track of those descent paths that leave the set $O(K)$. However, the pseudo-probabilities are calculated for all of the observed data just like in the basic algorithm.

As a result we hope to get a consistent configuration for the component K and then fix some of the acquired data (e.g. on the set P_k) and run the algorithm again in its basic form.

There is no guarantee that a configuration achieved by extracting a component is consistent with the whole data, but in practice this often seems to be the case. Our implementation APE can extract any number of components at the same time.

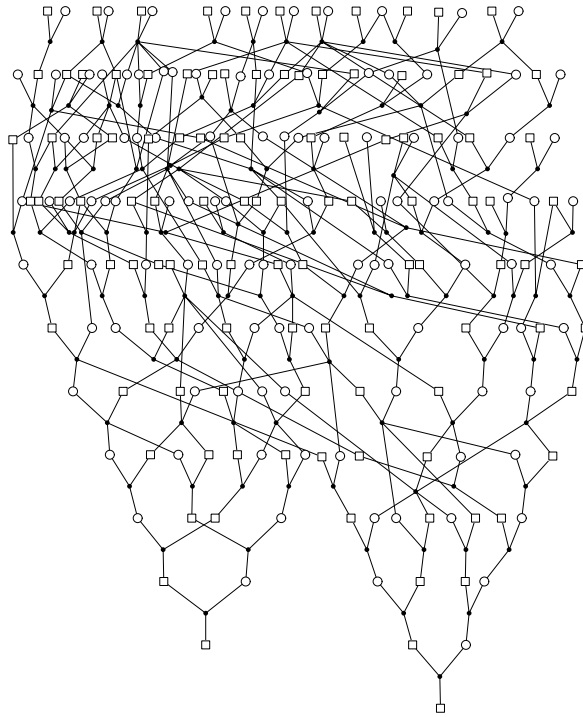
4 Results

APE was tested in three different settings. All tests were run under the Linux operating system on Intel CPU architecture. In the first two tests we used a 450 Mhz processor with 128 Mb of RAM, whereas the third test was run on a 2800 Mhz processor with 900 Mb of RAM. In all three settings genetic data were simulated by picking founder alleles from the uniform distribution, dropping the alleles down through the pedigree and finally considering only some part of the obtained total configuration as known data. In all tests the reported allele numbers refer to the actual data sets that were analysed by APE, not to the number of founder alleles which were usually slightly larger, since not all the founder alleles remained in the observed data. We also note that the pedigrees we have used are connected and cannot be divided into smaller parts without breaking some relationships between individuals.

As the first test we used the same pedigree (221 individuals, 12 levels) as Luo and Lin [12]. The pedigree comes from the Hutterite population living in North America and traces the ancestors of two infantile hypophosphatasia (HOPS) affected individuals back to 48 founders (Figure 1). This is an example of pedigrees that do not have well-defined generations.

We analysed 1000 data sets for each of the 24 combinations of the ratio of known geno-

Figure 1: HOPS pedigree.



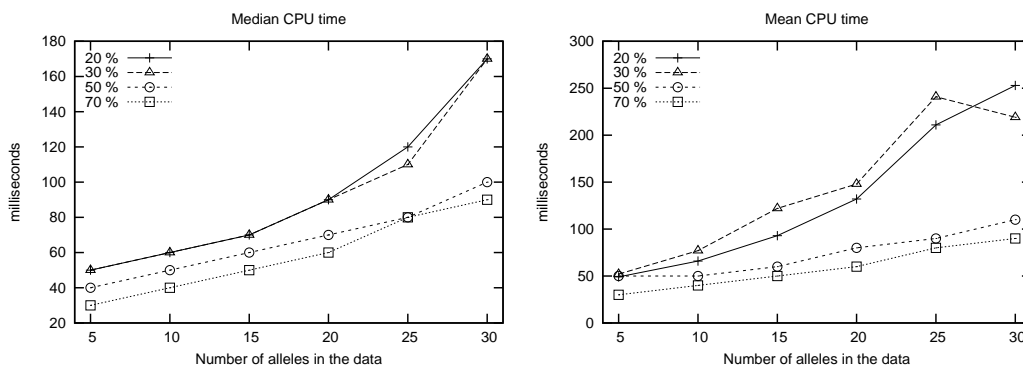
types (20%, 30%, 50% or 70%) and the number of alleles in the data (5,10,15,20,25,30).

Our strategy was to use the parameters ($M_{clu} = 10, M_{comp} = \infty, M_{lev} = \infty, \kappa = 3.0, t = 1.0$) and automatically alternate between the basic algorithm and the extraction of components with the upper bound of 10 extractions per data set. Of all the 24,000 data sets this was enough in all but 6 cases in which we had to change the parameters κ and M_{clu} before a consistent configuration was found. The reported running times include all the CPU time APE took from the first attempt with the basic algorithm to the moment when a configuration was found. Only three sets took more than 10 seconds and the longest running time, 32 seconds, was encountered with 25 alleles and 30% of the data observed.

Comparing our results (Figure 2) with those in [12] shows that our median times grow much slower with respect to the number of alleles. In [12] START is tested on a 1800

Mhz processor and the median time increases from less than a second (5 alleles) up to 120 seconds (20 alleles) when 30% of data is known. For APE the corresponding median times are 0.05 (5 alleles) and 0.09 (20 alleles) seconds, obtained on a 450 Mhz processor. However, since different computers are used, the running times may not be comparable in a straightforward way.

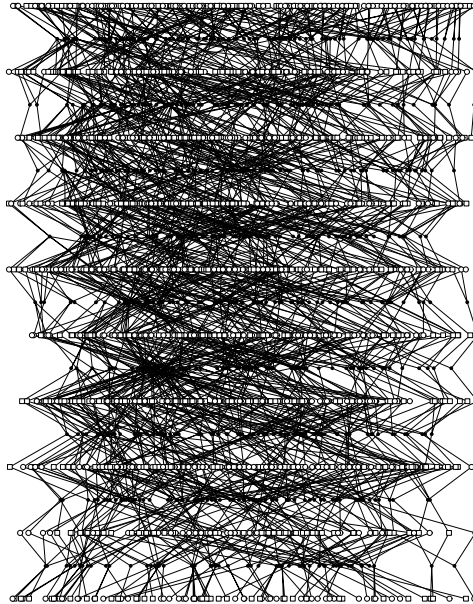
Figure 2: Mean and median running times on HOPS pedigree with different amounts of observed data.



In the second test we considered larger pedigrees. The pedigrees were simulated using the method introduced by Gasbarra et al. [3] in which the ancestry of a certain number of individuals belonging to the same population is traced backwards in time. The mating parameters defined in [3] were set to $\alpha = 1.0$ and $\beta = 0.001$ as to impose a certain degree of monogamy in the pedigrees. The base population was small and constant sized containing 400 individuals in each generation (200 males and 200 females). The number of generations was set to 10 and the first generation in the pedigrees consisted of 80 individuals. These choices of parameters created pedigrees with an average size of 1500 individuals, and the eldest generations contained about 170 founders. The proportion of nuclear families was about 70% and on average a component consisted of 2.4 parents and 2.2 children. Approximately 40% of the individuals were inbred i.e. their parents were related within the pedigree. This shows that these pedigrees contain many inbreeding loops and are thus hard to handle with exact methods such as peeling. One of these

pedigrees is shown in Figure 3.

Figure 3: An example of the pedigrees of the second test case.

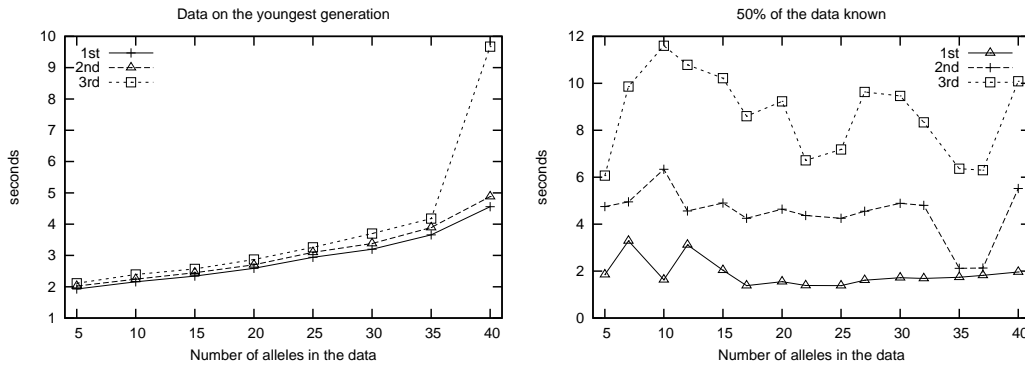


Again we simulated the data from the uniform distribution for founders and dropped the alleles down the pedigrees. We considered two different forms of given data. In the first version the data were available only on the youngest generation whereas in the second version 50% of the individuals randomly chosen from the whole pedigree, were genotyped. The basic set of parameters were $M_{clu} = 10$, $M_{comp} = \infty$, $M_{lev} = \infty$, $t = 1.0$, and for κ the values 1.0 and 100.0 were frequently used.

In the first version we considered 8 and in the second version 15 different numbers of alleles within the range from 5 to 40. For both variants and each number of alleles, 100 data sets were analysed and the results in the form of the first, the second (i.e. the median) and the third quartiles (i.e. the 25th, 50th and 75th smallest CPU times, respectively) are illustrated in Figure 4.

These results also show that the median running times of APE are only slightly affected by the number of alleles. However, in the case where the data are observed only on the

Figure 4: The quartiles of the running times in the second test.



lowest level of the pedigree, the third quartile strongly increases when the number of alleles is raised from 35 to 40. At this point the number of more difficult cases increases even though the median time still grows only mildly. With the first version of the test three data sets (out of 800) took more than 2 minutes, and the maximum running time was 2 minutes 51 seconds, encountered with 40 alleles.

Interestingly in the second version of the test, where the data are distributed all over the pedigree, the most complex data sets were encountered with 10 alleles. This is because increasing variability at the upper levels of the pedigree gives more information in the form of the pseudo-probability model and thus may make things easier for APE. In the second version there were 4 cases (out of 1500) that took over 3 minutes, the maximum running time being 8 minutes 37 seconds.

To ensure that our ordering process enhances the algorithm, we tried also those data sets with 10 alleles and known data only on the youngest generation with a version that did not order configurations at all. In that case only one of the 100 data sets was completed with the same reduction parameters that resulted in 97 successes when the ordering method was used. (In the remaining 3 cases we had to extract components.)

The purpose of the third test case was to see how APE copes with even larger pedigrees. Here we simulated with the method of Gasbarra et al. [3] a pedigree that extended over fifteen generations. The pedigree was embedded in a population that started with 1000

individuals at the eldest generation (level 14) and grew exponentially by the factor 1.2 at each generation resulting in a population size of 12,800 at the youngest generation (level 0). Each generation of the population contained an equal number of males and females. The parameters governing the mating behaviour were adjusted to $\alpha = 0.1$ and $\beta = 0.0001$ so as to make the setting more realistic by imposing a certain degree of monogamy in the pedigree. The pedigree simulation tracked the origins of 50 individuals belonging to the youngest generation of the described population to the founder generation. The pedigree contained 4921 individuals divided into 2250 components of which 1949 (87 %) were nuclear families (i.e. two parents and their common children). Of the remaining components the largest one contained 31 individuals (7 parents and 24 children). The average component size over the whole pedigree was 2.2 parents and 2.1 children, and the ratio of inbred individuals (with respect to the pedigree) was 75%. The pedigree contained 159 founders, all at the eldest generation.

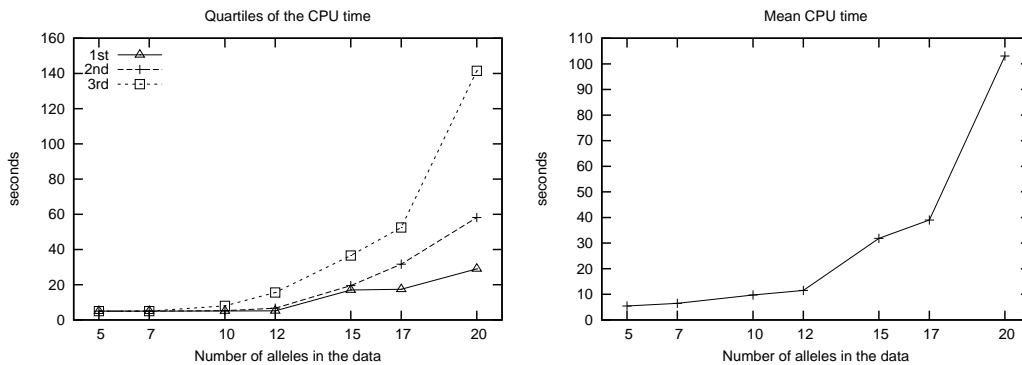
The data were simulated again by sampling founder genotypes from the uniform distribution of alleles, dropping the alleles down to the youngest generation, and giving only some part of the resulting total configuration as an input to APE. We considered situations where the data were available only on the four youngest generations. Namely the levels 0,1,2 and 3 were 100%, 75%, 50% and 25% typed, respectively, and the rest of the pedigree was left untyped. The typed individuals at levels 1,2 and 3 were chosen randomly. This procedure set the fraction of known data to about 7%.

We used seven different numbers of alleles (5, 7, 10, 12, 15, 17, 20) and for each of them we simulated 100 configurations that APE examined. The default parameters were ($M_{clu} = 2, M_{comp} = \infty, M_{lev} = 2, \kappa = 15.0, t = 1.0$), usual changes being $M_{clu} = 5, 10, \kappa = 10.0, 5.0$ and $t = 2.0$. Again our strategy was to alternate between the basic algorithm and the extraction of components, and if that did not yield a result, then change the value of κ and/or M_{clu} . Eventually, a consistent configuration was found in all 700 cases within 10 minutes. The statistics can be found in Figure 5.

In this large pedigree the number of alleles seems to have a stronger effect on running

times than in the other two settings. The increased complexity was also shown by the fact that in the 20-allele case we had to change parameters manually more often than before. Note that we used a different computer from the one used in the previous tests (because of the larger memory requirements), so the numerical values are not comparable.

Figure 5: The quartiles and the mean of running times with the 4921-individual pedigree.



We also tried START on the same computer on four randomly picked representatives of these data sets (Table II). For each set we executed a pilot run of 5,000 iterations to adjust the tolerance parameter s as suggested by Luo and Lin [12]. For the other parameters we used the default values ($H = 1,000$, $h = 10$, $T = 5$, $r = 0.1$) which according to Luo and Lin [12] have turned out to be good choices regardless of the pedigree structure, if the allele number is in the range 3-20. The number of iterations were adjusted so that the maximum allowed running time was about 10 hours.

For the data sets with 5 ($s = 10$) and 10 ($s = 20$) alleles START succeeded, but in the other two cases with 15 ($s = 45$) and 20 ($s = 65$) alleles START did not find a consistent configuration within 10 hours (380,000 iterations with 15 alleles and 200,000 iterations with 20 alleles). APE found consistent configurations for all four data sets within 45 seconds. However, some other parameter values for START might be better in these cases and due to the stochastic nature of START, it is possible that repeated runs could have yielded results also in the two more difficult cases within the given time limit.

In Table II we have also included the running times of PedCheck [16] that checks the

Mendelian consistency of the data by genotype elimination. However, PedCheck does not output a single configuration (as APE and START do) but only tells whether one exists. It also turned out that the peeling method used in the preparation step (prep) of the Loki program [6],[7] was unable to handle any of these data sets.

Table II: The running times in seconds of the four data sets on the 4921-member pedigree.

alleles	APE	START	PedCheck	prep
5	5	105	260	–
10	13	26640	480	–
15	22	–	1320	–
20	44	–	4920	–

5 Discussion

We have introduced a new method for extending partially known genotype data on the given pedigree to a complete genotype configuration. The method can be used in finding starting points for MCMC simulations as well as in trying to prove the consistency of data. Our method, implemented in the program APE, is very different from the other available program START [12] that addresses the same problem. START uses stochastic Gibbs sampling with relaxed penetrances and updates one individual at a time, whereas APE proceeds deterministically relying on the heuristic ordering method that has proven to work well in the tested cases.

In practice, it would be helpful if we could consider all linked loci together when generating a starting point for an MCMC sampler, since unrealistic recombination rates may cause the sampler to mix slowly. This issue discussed also by Heath [5] and Luo and Lin [12] has not been taken into account either in APE or in START. Straightforward generalizations that consider multiple loci at the same time would remarkably increase the

number of possibilities, huge already in the case of a single locus, and remain a question for further study.

Another practical question is whether we could reasonably conclude that the data contain Mendelian errors if APE fails to find a consistent configuration. In each particular case, we could compute an empirical success rate with a fixed set of parameters based on such simulated data sets that have the same form (i.e. the same pedigree, the same number of alleles and the same typed individuals) as the original data. The Bayesian posterior probability of an inconsistency would then require a quantification of our prior belief in the consistency of data. In the frequentist setting we could try to find a set of parameters for which the success rate is large enough, and if APE still fails on the original data, we could reject the hypothesis of data being consistent with the known significance level. Unfortunately these methods do not seem very practical, at least not in large and complex cases.

We note, however, that APE can spot the most evident Mendelian errors already by an initial check of the data and also some of more complex errors with the help of pseudo-probabilities explained in section 3.2.1. It may also be possible to restrict a complex error to a smaller part of the pedigree on which exhaustive search is possible.

The philosophy we have used in testing APE is to prefer short runs with significant restrictions ($M_{clu} = 2, 5, 10$) over a single long run. Our experience has shown that suitable parameters and/or component extractions result in situations where this strategy yields results fast. In complex (but consistent) cases it may take many attempts with different parameters before a configuration is found. This sensitivity to parameter values may be considered a drawback of APE and makes it difficult to do large-scale testing on still more complex data sets than those explained in Results, since it is more difficult to automate the choice of parameters and extractions and some human intervention is more often required.

In addition to the restriction parameters there are also the ordering parameters κ and t . The parameter κ controls the interplay of pseudo-probabilities and variation coefficients

in the ordering process. In short: larger κ makes the variation coefficients more significant. Especially in cases where there is little data on upper levels of the pedigree, large values of κ may be preferable in the first run. The default value for the exponent t is 1.0, but in some complex cases we have found larger values like 2.0 to be better choices.

Software

APE is written in C-language and was tested under the Linux operating system on Intel CPU architecture. The compiler was GCC 2.95.4. The source code, as well as some of those data sets we used in testing, are available at the web page www.helsinki.fi/~mpirinen/download.

Acknowledgments

We would like to thank Elja Arjas, Mikko Sillanpää, Heikki Mannila, Esko Ukkonen and Simon Heath for useful comments. The pedigree figures are drawn with Pedfiddler by J C. Loredó-Osti and Kenneth Morgan. This work was supported by the research grant 53297 from the Academy of Finland.

References

- [1] Aceto L, Hansen J, Ingólfssdóttir A, Johnsen J, Knudsen J. 2004. The complexity of checking consistency of pedigree information and related problems. *JCST* 19(1): 42-59.
- [2] Elston R, Stewart J. 1971. A general model for the genetic analysis of pedigree data. *Hum Hered* 21: 523 - 542.
- [3] Gasbarra D, Sillanpää M, Arjas E. 2005. Backward simulation of ancestors of sampled individuals. *Theor Popul Biol* 67: 75-83.
- [4] Geman S, Geman D. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans Pattern Anal* 6: 721-741.
- [5] Heath S.C. 1998. Generating consistent genotypic configurations for multi-allelic loci and large complex pedigrees. *Hum Hered* 48: 1-11.
- [6] Heath SC. 1997. Markov chain Monte Carlo segregation and linkage analysis for oligogenic models. *Am J Hum Genet* 61: 748 - 760.
- [7] Heath SC, Snow GL, Thompson EA, Tseng C and Wijsman EM. 1997. MCMC segregation and linkage analysis. *Genet Epidemiol* 14: 1011 - 1015.
- [8] Lange K. 2002. *Mathematical and statistical methods for genetic analysis*. 2nd edition, Springer-Verlag, New york, pp 82-83.
- [9] Lange K, Goradia TM 1987. An algorithm for automatic genotype elimination. *Am J Hum Genet* 40: 250 - 256.
- [10] Lauritzen S, Sheehan NA. 2003. Graphical models for genetic analyses. *Stat Sci* 18: 489-514.

- [11] Lin SL, Thompson EA, Wijsman E. 1993. Achieving irreducibility of the Markov chain Monte Carlo method applied to pedigree data. *IMA J Math Appl Med Biol* 10: 1-17.
- [12] Luo Y, Lin S. 2003. Finding starting points for Markov chain Monte Carlo analysis of genetic data from large and complex pedigrees. *Genet Epidemiol* 25: 14 - 24.
- [13] Mackworth A. 1987. Constraint satisfaction. *Encyclopedia of Artificial Intelligence* vol 1, Shapiro S (editor). John Wiley & Sons. pp 205 - 211.
- [14] Marriot K, Stuckey P. 1998. *Programming with constraints: An introduction*. The MIT Press. ch. I.3.
- [15] O'Connell J, Weeks D. 1999. An optimal algorithm for automatic genotype elimination. *Am J Hum Genet* 65: 1733 - 1740.
- [16] O'Connell J, Weeks D. 1998. PedCheck: A program for identification of genotype incompatibilities in linkage analysis. *Am J Hum Genet* 63: 259 - 266.
- [17] Sheehan NA. 2000. On the application of Markov chain Monte Carlo methods to genetic analyses on complex pedigrees. *Int Stat Rev* 68: 83 - 110.
- [18] Thompson EA. 2000. Statistical inference from genetic data on pedigrees. NSF-CBMS Regional conference series in probability and statistics Volume 6.

Appendix A: The algorithm of APE

```

l:=0;

do
{
  find and save all the configurations for Pl based on the observed
  data and the current configurations for P0, ..., P{l-1};

  if configurations for Pl exist
  {
    order the configurations for Pl and choose
    the first of them as the current configuration for Pl;
    l:=l+1;
  }
else
{
  if there is a 0<k<l s.t. there are unchecked
  configurations left for Pk
  {
    choose the largest such k and the first unchecked configuration
    for Pk as the current configuration for Pk;
    l:=k+1;
  }
else
  l:=0;
}
}
while(0<l<L)

if(l=L)
  an admissible configuration is found
if(l=0)
  data are inconsistent with the pedigree

```

Biographies

Matti Pirinen received his M.Sc. degree in mathematics from the University of Helsinki in 2004 and is currently a PhD-student at the university's Department of Mathematics and Statistics. He works with the Centre of Population Genetic Analysis, chosen a Centre of Excellence by the Academy of Finland for the period 2002-2007. Pirinen's PhD studies are also partly funded by the ComBi graduate school. His research interests include mathematical and statistical genetics and related computational aspects.

Dario Gasbarra received his M.Sc. degree in mathematics at the University of Rome "La Sapienza" in 1991, and the Ph.D. in applied mathematics at the University of Oulu in 1998. He is currently based at the Department of Mathematics and Statistics of the University of Helsinki, working as a researcher in the Centre of Population Genetic Analysis, funded by the Academy of Finland. His research interests are bioinformatics, mathematical and computational statistics, and stochastic analysis.