BMC Bioinformatics

Research

Estimate haplotype frequencies in pedigrees

Qiangfeng Zhang, Yuzhong Zhao, Guoliang Chen* and Yun Xu

Address: Department of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, 230027, China

Email: Qiangfeng Zhang - qfzhang@mail.ustc.edu.cn; Yuzhong Zhao - zyzh@mail.ustc.edu.cn; Guoliang Chen* - glchen@ustc.edu.cn; Yun Xu - xuyun@ustc.edu.cn

* Corresponding author

from Symposium of Computations in Bioinformatics and Bioscience (SCBB06) in conjunction with the International Multi-Symposiums on Computer and Computational Sciences 2006 (IMSCCS|06)
Hangzhou, China. June 20–24, 2006

Published: 12 December 2006

BMC Bioinformatics 2006, **7**(Suppl 4):S5 doi:10.1186/1471-2105-7-S4-S5

Abstract

Background: Haplotype analysis has gained increasing attention in the context of association studies of disease genes and drug responsivities over the last years. The potential use of haplotypes has led to the initiation of the HapMap project which is to investigate haplotype patterns in the human genome in different populations. Haplotype inference and frequency estimation are essential components of this endeavour.

Results: We present a two-stage method to estimate haplotype frequencies in pedigrees, which includes haplotyping stage and estimation stage. In the haplotyping stage, we propose a linear time algorithm to determine all zero-recombinant haplotype configurations for each pedigree. In the estimation stage, we use the expectation-maximization (EM) algorithm to estimate haplotype frequencies based on these haplotype configurations. The experiments demonstrate that our method runs much faster and gives more credible estimates than other popular haplotype analysis software that discards the pedigree information.

Conclusion: Our method suggests that pedigree information is of great importance in haplotype analysis. It can be used to speedup estimation process, and to improve estimation accuracy as well. The result also demonstrates that the whole haplotype configuration space can be substituted by the space of zero-recombinant haplotype configurations in haplotype frequency estimation, especially when the considered haplotype block is relatively short.

Background

The modelling of human genetic variation is critical to the understanding of the genetic basis for complex diseases. Single nucleotide polymorphisms (SNPs) are the most frequent form of variation. The Human Genome Project and other large-scale efforts have identified millions of SNP markers. Although each marker can be analyzed independently, it is more informative to analyze them in groups. Therefore, it is useful to analyze haplotypes (haploid genotypes), which are sequences of linked markers on a single chromosome. In diploid organisms, such as human beings, chromosomes come in pairs, and experi-

ments often yield genotype information, which blend haplotype information for chromosome pairs. There is growing evidence that, in order to better characterize the role of a candidate gene, full haplotype information should be exploited instead of using only genotype information. Unfortunately, it is both time-consuming and expensive to derive haplotype information experimentally. This explains the increasing interest in inferring haplotype information, or haplotyping, computationally. In fact, the potential use of haplotypes has led to the initiation of the HapMap project which is to investigate haplotype patterns in the human genome in different populations. Haplotype inference and frequency estimation are essential components of this endeavour.

Genotype data can be with or without any pedigree information, the first category is called population genotype data while the second one is pedigree genotype data. A large number of algorithms have been designed to estimate haplotype frequencies based on population data [1-4]. Among them, EM algorithms are most popular due to their interpretability and stability.

For any given pedigree genotype data, we can certainly discard the pedigree information and simply take the genotype sequences as the input of EM estimation algorithms for population data. However, it is well accepted that information obtained by analyzing pedigree genotype data is more reliable: the constraint provided by other members in a pedigree would force one genotype to settle on a unique haplotype pair as being most probable.

Here we propose a two-stage method to estimate haplotype frequencies in pedigrees. The first stage is the haplotyping stage, which finds out all feasible haplotype configurations for each pedigree. In the second estimation stage, we use EM algorithm to estimate haplotype frequencies in pedigrees based on the haplotype configurations inferred in the former stage.

In general, haplotyping pedigrees need consider the entire solution space of all possible consistent haplotype configurations. However, the genomic DNA can be partitioned into long blocks such that recombinations within each block are rare or even nonexistent [5,6]. Thus it is believed that haplotype configurations with fewer recombinations should be preferred in haplotype inference [7-9]. When the region of interest is so small that the expected number of recombinations in the pedigree data is very close to zero, the solution space of all consistent haplotype configurations can be replaced by that of zero recombination (provided it is non-empty) to estimate haplotype frequencies. It is because the contribution of the solutions of recombinations to the overall likelihood becomes so small compared to those of zero recombination while they bring considerable complexity to the computation. Thus, we are interested in finding the consistent haplotype configurations of zero recombination.

Wijsman [10] proposed a 20-rule algorithm, and O'Connell [11] described a genotype-elimination algorithm, both of which can be used to find out zero-recombinant haplotype configurations for pedigrees. Recently, Li and Jiang [8,9] showed that it could be solved in polynomial time. Here we propose an algorithm to find out zero-recombinant haplotype configurations in linear time using a technique called HCL-linkage analysis.

In the second stage, we use the EM algorithm to estimate haplotype frequencies based on haplotype configurations obtained from the haplotyping stage. We employ the Hardy-Weinberg Equilibrium to obtain the probabilities of founder genotypes and use a genetic model [12] to deduce the transmitted probabilities of non-founders. While the likelihood of each configuration is computed by multiplying the probabilities of each genotype, the frequency of each haplotype that appears in the configuration is calculated by a gene-counting method.

We implement all the algorithms in a C software package named HANAP (Haplotype ANAlysis in Pedigrees) and test its effectiveness and efficiency both on simulated and real data sets. The experimental results show that, our method runs much faster than the direct frequency estimation software that discards the pedigree information. Moreover, because our method utilizes such information, the estimation is more reliable.

Methods

Haplotyping stage: haplotyping algorithm based on HCL-linkage analysis

Excoffier's EM algorithm was widely applied in haplotype analysis [14,15]. Unfortunately, it should calculate the frequencies of all possible haplotype pairs consistent to each given genotype, which is unbearable in storage when the haplotype length grows to more than 20 [16]. O'Connell [10] showed that genetic information from relatives could be used to resolve one genotype's ambiguity, and thus reduce the number of haplotypes that should be considered. However, O'Connell's method had an exponential time complexity. Recently, Li and Jiang [8,9] showed that, for any genotype in a given pedigree, its ambiguity could be solved in cubic time ($O(m^3n^3)$), where n is the number of members in the pedigree and m is the number of loci in each genotype. Here we present a so-called HCL-linkage analysis method to do haplotyping in linear time ($O(mn)$).

HCL-linkage definition

Trios are simple pedigrees that contain only a pair of parents and a child. A consistent zero-recombinant haplotype configuration for a general pedigree should also be a consistent zero-recombinant haplotype configuration when restricted to each trio in this pedigree. Given trio $T = (F, M, C)$, here F is the father, M is the mother and C is the child, suppose that locus i of F, M, and C have alleles $\{a, b\}$, $\{c, d\}$ and $\{e, f\}$ (note that $\{e, f\} \subset (\{a, b\} \cup \{c, d\})$). The genotype information of C can be homozygous or heterozygous. If it is homozygous ($e = f$), then it is clear that the paternal allele and the maternal allele are the same (e or f). The situation becomes complicated if it is a heterozygous site ($e \neq f$). Table 1 lists out all possible situations. We can see that given the locus genotype information for the three members, it may or may not be possible to determine the paternal allele and the maternal allele for the child. We call a locus *ambiguous* if its inheritance relationship cannot be resolved.

In fact, for any trio, ignoring the ambiguous loci, the consistent (partial) haplotype configuration for the *unambiguous* loci is unique and specifies a linkage of alleles on some heterozygous loci for each node in the trio. We define such linkage as HCL-linkages (linkages of Haplotype Configuration on the non-ambiguous Loci).

Table 1: Imputing the paternal allele and the maternal allele for the child at a single locus

Conditions	Paternal	Maternal
$e = f$	e	f
$e \neq f$, $a = b$		
$e = a$	e	f
$f = a$	f	e
$e \neq f$, $a \neq b$, $c = d$		
$e = c$	f	e
$f = c$	e	f
$e \neq f$, $a \neq b$, $c \neq d$		
$a = c$, $b = d$	*Ambiguous*	
$a = c$, $b \neq d$		
$e = b$ or $f = d$	e	f
$e = d$ or $f = b$	f	e
$a \neq c$, $b = d$		
$e = a$ or $f = c$	e	f
$e = c$ or $f = a$	f	e
$a = d$, $b = c$	*Ambiguous*	
$a = d$, $b \neq c$		
$e = b$ or $f = c$	e	f
$e = c$ or $f = b$	f	e
$a \neq d$, $b = c$		
$e = a$ or $f = d$	e	f
$e = d$ or $f = a$	f	e
$a \neq c \neq b \neq d$		
$e = a$ or $e = b$	e	f
$e = c$ or $e = d$	f	e

Definition

An **HCL-linkage** *ψ is a quadruplet <v, RE, LS, PH> defined on node v and specified by the unique consistent (partial) haplotype configuration within a trio that contains v. Here v denotes the node to which the HCL-linkage belongs. $LS = \{a_1,...,a_l\}$ is the set of heterozygous loci where the haplotype configuration has been inferred. $PH = \{ph, ph'\} = \{(h_1...h_l), (h_1'...h_l')\}$ records the two (partial) haplotypes imputed on these loci. $RE = (R, R')$ denotes that ph (respectively ph') is inherited from or will be passed on to the node in set $R(R')$.*

An HCL-linkage describes the partial haplotype configuration of a node and the inheritance relationship between the parents and their children. Under our definition, we can conclude that every haplotype configuration should be consistent with any HCL-linkage specified by each trio in the pedigree.

Merge and transfer operations over HCL-linkages

In the case of multiple generations and multiple children, loci on one node may be linked by different HCL-linkages. HCL-linkages of the same node should be merged if they can. There are three cases when merging two HCL-linkages $\psi_1 = <v, (R_1, R_1'), LS_1, \{ph_1, ph_1'\}>$ and $\psi_2 = <v, (R_2, R_2'), LS_2, \{ph_2, ph_2'\}>$ on node v.

Case (1): $(R_1 \cup R_1') \cap (R_2 \cup R_2') \neq \Phi$

l.a) $R_1 \cap R_2 \neq \Phi$ or $R_1' \cap R_2' \neq \Phi$, it means that both ph_1 and ph_2 are from the nodes in R_1 and R_2 said ph_1' and ph_2' are from the nodes in R_1' and R_2', so ph_1 and ph_2 should be on the same haplotype, and ph_1' and ph_2' on the other: i) $LS_1 \cap LS_2 = \Phi$, or $LS_1 \cap LS_2 \neq \Phi$ but ph_1 equals ph_2 when restricted to loci in $LS_1 \cap LS_2$, it means that ψ_1 and ψ_2 are *compatible*. In this case, they should be merged to $\psi = <v, (R_1 \cup R_2, R_1' \cup R_2'), LS_1 \cup LS_2, \{ph_1 \cup ph_2, ph_1' \cup ph_2'\}>$, here $ph_1 \cup ph_2$ denote a longer partial haplotype, which alleles equal to those of ph_1 and ph_2 when restricted to loci in LS_1 and LS_2; ii) $LS_1 \cap LS_2 \neq \Phi$ and ph_1 doesn't equal ph_2 when restricted to $LS_1 \cap LS_2$, it means that ψ_1 and ψ_2 are *incompatible*, *i.e.* no haplotype configuration can satisfy the two HCL-linkages in the same time.

1.b) $R_1 \cap R_2' \neq \Phi$ or $R_1' \cap R_2 \neq \Phi$, it means that ph_1 and ph_2' should be on the same haplotype, and ph_1' and ph_2 on the other. Similarly, ψ_1 and ψ_2 can be merged to $\psi = <v, (R_1 \cup R_2', R_1' \cup R_2), LS_1 \cup LS_2, \{ph_1 \cup ph_2', ph_1' \cup ph_2\}>$ when they are *compatible*.

Case (2): $(R_1 \cup R_1') \cap (R_2 \cup R_2') = \Phi$, but $LS_1 \cap LS_2 \neq \Phi$,

2.a) ph_1 equals ph_2 (and ph_1' equals ph_2' consequently) or ph_1 equals ph_2' (then ph_1' equals ph_2) when restricted to $LS_1 \cap LS_2$, it means that ψ_1 and ψ_2 are *compatible*, in this case, they should be merged to $\psi = <v, (R_1 \cup R_2, R_1' \cup R_2'),$

$LS_1 \cup LS_2$, $\{ph_1 \cup ph_2, ph_1' \cup ph_2'\}>$ or $\psi = <v, (R_1 \cup R_2', R_1' \cup R_2), LS_1 \cup LS_2, \{ph_1 \cup ph_2', ph_1' \cup ph_2\}>$.

2.b) Else, ph_1 doesn't equal ph_2 or ph_2' when restricted to $LS_1 \cap LS_2$, it means that ψ_1 and ψ_2 are *incompatible*.

Case (3): $(R_1 \cup R_1') \cap (R_2 \cup R_2') = \Phi$, and $LS_1 \cap LS_2 = \Phi$,

In this case, ψ_1 and ψ_2 cannot be merged and both should be recorded in a HCL-linkage set Ψ_v for node v.

With the merge operation, we can define the normalizing of a set of HCL-linkages Ψ_v: *normalizing* a set Ψ_v of HCL-linkages on node v means repeatedly applying the merge operation for pairs of HCL-linkages in Ψ_v until, $\forall \psi_i, \psi_j \in \Psi_v$, $(R_i \cup R_i') \cap (R_i \cup R_i') = \Phi$, and $LS_1 \cap LS_2 = \Phi$. Ψ_v is then said to be *normalized*. From now on, if there is no further notice, Ψ_v should be normalized after any changes.

Like genetic information, HCL-linkages will be passed on from generations to generations. Without loss of generality, let us define the transfer of HCL-linkage information from child C to its parent F. The other case from F to C would be similar. Let Ψ_C and Ψ_F represent the normalized HCL-linkage sets of C and F respectively, and let HS be the set of homozygous loci of F. The transfer of Ψ_C from C to F results in changes to Ψ_F, where each $\psi_C = <C, (R_C, R_C'), LS_C, \{ph_C, ph_C'\}> \in \Psi_C$ is transferred independently. There are two cases to consider.

Case (1): if $F \in R_C$ (or respectively, $F \in R_C'$), add $\psi_F = <F, (\{C\}, \Phi), LS_C - HS, \{ph_F, ph_F'\}>$ to Ψ_F, here ph_F equals the resulting partial haplotypes of ph_C (respectively ph_C') when restricted to loci in $LS_C - HS$ and ph_F' is the compensatory partial haplotypes of ph_F consistent to genotype g_F.

Case (2): else, $F \notin R_C \cup R_C'$: i) both ph_C and ph_C' are consistent with the partial genotype g_F when restricted to loci in LS_C, then add $\psi_F = <F, (\Phi, \Phi), LS_C - HS, \{ph_F, ph_F'\}>$ to Ψ_F, here ph_F and ph_F' equal the resulting partial haplotypes *of* ph_C and ph_C' when restricted to loci in $LS_C - HS$; ii) ph_C' (respectively ph_C) is not consistent with the partial genotype g_F when restricted to loci in LS_C, then add $\psi_F = <F, (\{C\}, \Phi), LS_C - HS, \{ph_F, ph_F'\}>$ to Ψ_F, here ph_F equals the resulting partial haplotypes of ph_C (respectively ph_C') when restricted to loci in $LS_C - HS$ and ph_F' is the compensatory partial haplotypes of ph_F consistent to genotype g_F. Note that at least one of ph_C and ph_C' should be consistent with the partial genotype g_F.

Remember that Ψ_F should be normalized whenever adding a new HCL-linkage to it. In the case of transferring an HCL-linkage ψ_F from F to C, resulting in adding $\psi_C = <C, (R_C, R_C'), LS_C, \{ph_C, ph_C'\}>$ to Ψ_C, note that we should add M into R_C' whenever we have determined that $F \in R_C$.

Our merge and transfer operations will not bring more or lose any HCL-linkage information for building consistent haplotype configurations.

Main HCL-linkages analysis haplotyping algorithm

Before the algorithm, we preprocess each trio in the pedigree. Whenever a trio specifies an HCL-linkage for node v, it will be stored in the HCL-linkage set Ψ_v. The objective of the algorithm is to collect the complete HCL-linkage information for each node, which is accomplished by traversing the tree twice.

Firstly, we will convert the input pedigree into a rooted searching tree **T** (at an arbitrary node R) (Step 1). Then we traverse **T** in post-order to transfer and merge the HCL-linkage information for each node from its relatives (Step 2). We do this from the left lowest nuclear family $\mathbf{F}_o$. The HCL-linkages in nuclear family $\mathbf{F}_o$ will be merged at both parents, and then be transferred to the root of the sub-tree. The same operations will be conducted in its parental nuclear family on HCL-linkages specified in this family as well as on those transferred from its child families. And at last, we collect all the HCL-linkages at the root R. In Step 3, we traverse **T** again in pre-order and transfer the linkage in another direction from R to its farmost descendants.

After step 3, the HCL-linkage set of each node preserves all HCL-linkages in the pedigree. In step 4, we choose a node v arbitrarily. Set Ψ_v contains several HCL-linkages $\psi_1, \psi_2, ..., \psi_l$ defined on disjoint locus set $LS_1, LS_2, ...LS_l$. When a set of loci are linked by one HCL-linkage, they can be viewed as a compound locus, and the two partial haplotypes can be viewed as two compound alleles. These "loci" (and "alleles") will be treated equally as the other heterozygous loci and homozygous loci that are not involved in any HCL-linkage. We arbitrarily select one allele from the two at each locus to form a haplotype; the other alleles form another haplotype. It is called an *imputing schema*. Whenever the haplotype configuration of one node is determined, it can be used to determine the configurations of its relatives, and those of the whole pedigree at last.

During our algorithm, Incompatibleness may occur when normalizing HCL-linkage set Ψ_v. Then we declare that there is no solution and exit from the algorithm immediately. Even in step 4, incompatibleness may still occur when applying the haplotypes of the parents to resolve the genotype of the children in the case that an individual node has multiple children. Figure 1 shows an example. The key point is, if it exists a consistent haplotype configuration for a nuclear family $(F, M, C_1, C_2, ..., C_d)$, every arbitrary imputing schema s can output one feasible solution ζ. Contrarily, if one imputing schema ends with incom-

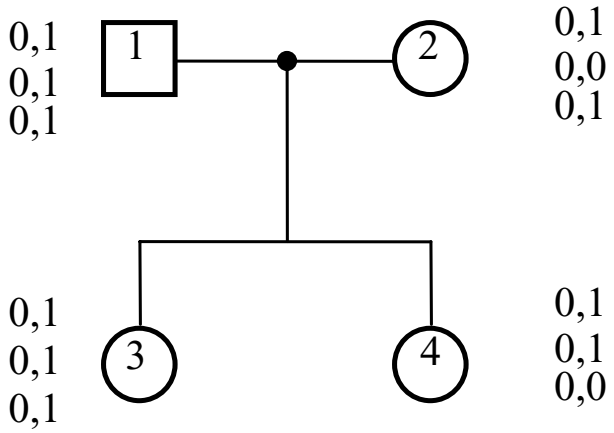

Figure 1
Incompatibleness when imputing alleles.

patibleness, other schemata will fail too. We will prove this in the appendix.

The time complexity and space complexity of our algorithm are both $O(mn)$ where n is the number of the members in the pedigree and m is the length of the loci.

Frequency estimation stage

Suppose that we are given K pedigrees $\mathbf{P} = \{P_1, P_2,...,P_K\}$. Each P_i consists of n_i nodes $v_{i,j}$ $(1 \le i \le K,\ 1 \le j \le n_i)$, in which the first n_i' are founders. The genotype of node $v_{i,j}$ $(1 \le j \le n_i)$ is $g_{i,j}$. Suppose that there are π_i consistent solutions for pedigree P_i and the s-th solution is: $S_{s,i} = <S_{s,i,1}, S_{s,i,2},..., S_{s,i,n_i}>$ $(1 \le s \le \pi_i)$, where $S_{s,i,j} = <\alpha_{s,i,j,1}, \beta_{s,i,j,2}>$ is a haplotype pair of genotype $g_{i,j}$. All haplotypes appear in these solutions form a list of haplotypes $H = \{h_1, h_2,...,h_l\}$ with frequencies $\Theta = \{\theta_1, \theta_2,...,\theta_l\}$, here $\theta_1 + \theta_2 + ... + \theta_t = 1$ is what we want to estimate.

The likelihood of haplotype frequencies given the observed pedigree data is,

$$\begin{aligned} L(\theta_1,\theta_2,...,\theta_l) &= \Pr(P_1, P_2,..., P_k \mid \Theta) \\ &= c \times \prod_{i=1}^{K} \Pr(P_i \mid \Theta) \\ &= c \times \prod_{i=1}^{K} (\sum_{s=1}^{\pi_i} \Pr(S_{s,i} \mid \Theta)) \end{aligned} \quad (2)$$

Under the assumption of random mating, the paternal haplotype configuration and the maternal haplotype configuration are independent, and the child's haplotype configuration is transmitted from its parents. We have:

$$L(\theta_1,\theta_2,...,\theta_l) = c \times \prod_{i=1}^{K} (\sum_{s=1}^{\pi_i} (\prod_{j=1}^{n_i'} \Pr(S_{s,i,j} \mid \Theta) \cdot \prod_{j'=n_i'+1}^{n_i} \Pr(S_{s,i,j'} \mid < S_{s,i,j'}^{F}, S_{s,i,j'}^{M} >))) \quad (3)$$

Here $\Pr(S_{s,i,j}|\Theta)$ is the probability of haplotype configuration of the founder nodes, it can be computed using the Hardy-Weinberg Equilibrium. $\Pr(S_{s,i,j'}| < S_{s,i,j'}^{F},\ S_{s,i,j'}^{m} >)$ is the gamete transmission probabilities of haplotype configuration $S_{s,i,j}$ with the parental haplotype configurations of $S_{s,i,j'}^{F}$ and $S_{s,i,j'}^{m}$. It can be computed using a genetic model presented by Elston and Stewart [6].

EM algorithm estimates the haplotype frequencies Θ starting with the initial arbitrary values $\Theta^{(0)} = \{\theta_1^{(0)}, \theta_2^{(0)},..., \theta_l^{(0)}\}$. These initial values are used as if they were the unknown true frequencies to estimate solution frequencies $\Pr(S_{s,i}|\Theta)$ (the expectation step). These expected solution frequencies are used in turn to estimate haplotype frequencies at the next iteration $\Theta^{(1)} = \{\theta_1^{(1)}, \theta_2^{(1)},..., \theta_l^{(1)}\}$ (the maximization step), and so on, until convergence is reached.

Suppose that in the r-th iteration, $\Theta = \Theta^{(r)}$ and we want to estimate $\Theta^{(r+1)}$. Then we have:

$$\tilde{\Pr}(S_{s,i} \mid \Theta)^{(r)} = \frac{1}{K} \cdot \frac{\prod_{j=1}^{n_i'} \Pr(S_{s,i,j} \mid \Theta)^{(r)} \cdot \prod_{j'=n_i'+1}^{n_i} \Pr(S_{s,i,j} \mid < S_{s,i,j}^{F}, S_{s,i,j}^{M} >)^{(r)}}{\sum_{s'=1}^{\pi_i} (\prod_{j=1}^{n_i'} \Pr(S_{s',i,j} \mid \Theta)^{(r)} \cdot \prod_{j'=n_i'+1}^{n_i} \Pr(S_{s',i,j} \mid < S_{s',i,j}^{F}, S_{s',i,j}^{M} >)^{(r)})} \quad (4)$$

Let $\delta_{i,j,t}$ be an indicator variable equalling the number of haplotype h_t appear in solution $S_{s,i}$. Then the haplotype frequencies can be computed using a gene-counting method,

$$\theta_t^{(r+1)} = \frac{1}{2} \cdot \sum_{i=1}^{K} \sum_{j=1}^{\pi_i} \delta_{i,j,t} \cdot \tilde{\Pr}(S_{s,i} \mid \Theta^{(r)}) \quad (5)$$

There are several ways to initialize the haplotype frequencies $\Theta = \{\theta_1, \theta_2,..., \theta_l\}$. For instance, the initial haplotype frequencies can be chosen at random, or all haplotypes are equally frequent, *i.e.* $\Theta_t^{(0)} = 1/l$ $(t = 1, 2,..., l)$. Or that all initial haplotype frequencies are equal to the product of the corresponding single-locus allele frequencies (*i.e.*, a complete linkage equilibrium). Also, we can set all feasible solutions for each pedigree to be equally likely, *i.e.* $\Pr(S_{s,i}|\Theta^{(0)}) = 1/\pi_i$, $(j = 1,2,..., \pi_i)$. We can even initialize the haplotype frequencies by counting their occurrence in all the feasible solutions. Since in practical applications the EM algorithm could be trapped in some local maximum, we recommend to restart the algorithm several times with

different initial haplotype frequencies and better with a randomized additive perturbation.

The stopping (convergence) criterion is defined as the absolute value of the difference of Θ between consecutive iterations being less than some small value $\varepsilon > 0$.

Results

Simulated data set

In order to generate a pedigree genotype data set for simulation experiments, we generate a population of haplotypes H^* first, where each locus of each haplotype is set to some allele according to the probability distribution function P. In our simulation, we generated haplotypes of SNP loci as well as haplotypes of micro-satellite loci. For a biallelic SNP locus i, suppose that i happens to be one state with a probability of p_i, and to be the other state with a probability of $(1 - p_i)$. For a micro-satellite loci, suppose it has w different alleles: $a_1, a_2,...,a_w$, each appears with the probability of $p_1, p_2,...,p_w$ $(p_1 + p_2 +...+ p_w = 1)$.

Each founder node in any tested pedigree is arbitrarily assigned a pair of haplotypes according to their frequencies θ^*. The two haplotypes of a non-founder node are arbitrarily selected from those of its parents (one from the father, one from the mother). At last, the pair of haplotypes of the same node is blended to form a genotype corresponding to that node.

All experiments are conducted on a Windows server with 1.7G Hz CPU and 256 MB RAM. And for each parameter setting, 100 copies are randomly generated and the performance is evaluated by computing the average numbers in these 100 runs.

Running time of the haplotyping algorithm

One of the main contributions of our paper is to do haplotyping in linear time, so we firstly examine the running time with respect to different number of nodes of each pedigree (n) and different number of loci in each sequence (m).

Several different tree pedigree structures are used in the simulation, the first pedigree is Figure 1 in [15], which is a tree with 13 nodes. The second one is Figure 8 in [9], which is a tree with 29 nodes. The third one is a 21 node pedigree from Figure 5 of [15]. The results are given in Figure 2. It is obvious that our HCL-analysis haplotyping algorithm runs in linear time and thus could be applied to large-scale haplotype analysis.

Number of solutions

We compare the numbers of haplotypes that should be considered in the estimation stage, with and without the haplotyping stage. In our experiment, we set P_1 ($p_1 = p_2 =$ 0.5) and P_2 ($p_1 = 0.9$, $p_2 = 0.1$) for SNP loci, and set $w = 4$, P_3 ($p_1 = p_2 = p_3 = p_4 = 0.25$) and P_4 ($p_1 = 0.5$, $p_2 = p_3 = 0.2$, $p_4 = 0.1$) for micro-satellite loci. We let $|H^*| = 20$, and $\theta_1^* = 0.2$, $\theta_2^* = \theta_3^* = \theta_4^* = 0.1$, $\theta_5^* = \theta_6^* = \theta_7^* = \theta_8^* = 0.05$, $\theta_9^* = \theta_{10}^* = ... = \theta_{20}^* = 0.025$.

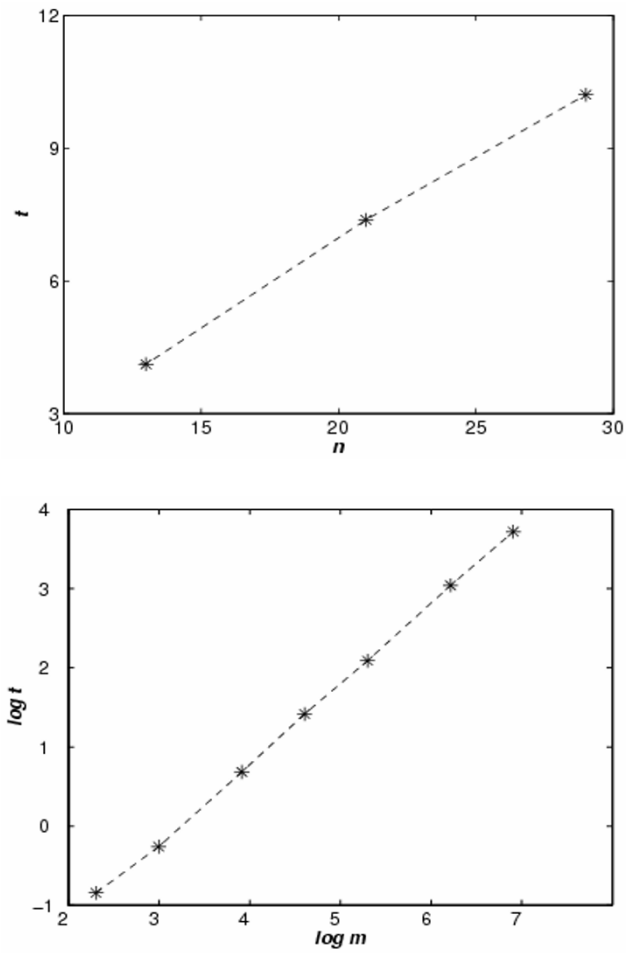

Figure 2
Running time of the haplotyping algorithm: (a) running time *vs*. number of nodes (m = 100); (b) running time *vs*. number of loci (n = 13).

When only trio pedigrees are considered, the average numbers of haplotypes are recorded in Table 2. We can see from the table that the numbers of haplotypes that should be estimated have been greatly reduced after the haplotyping stage (HANAP *vs.* directly), which will immediately bring the improvement on the running time.

We also consider a more complex pedigree that contains 13 nodes (Figure 1 of [15]). The average numbers of haplotypes are recorded in Table 3. We find that the number of haplotypes that should be estimated is even much

Table 2: Comparison of number of haplotypes (|*H*|) on trio pedigrees

Parameter settings (*m*, *k*)	directly				HANAP			
	P_1	P_2	P_3	P_4	P_1	P_2	P_3	P_4
(20, 20)	3.18e4	3.43e2	2.06e6	1.02e6	5.83e2	72.2	1.12e2	76.4
(20, 100)	1.49e5	1.68e3	1.02e7	5.02e6	2.67e3	3.55e2	5.72e2	3.61e2
(20, 200)	2.56e5	3.30e3	2.02e7	0.98e7	4.45e3	5.96e2	1.14e3	6.02e2
(100, 20)	N/A	8.13e6	N/A	N/A	5.90e5	2.60e2	6.91e2	2.74e2
(100, 100)	N/A	1.62e7	N/A	N/A	2.68e6	1.31e3	3.42e3	1.52e3
(100, 200)	N/A	3.21e7	N/A	N/A	4.65e6	2.55e3	6.91e3	2.89e3
(200, 20)	N/A	N/A	N/A	N/A	N/A	7.92e2	6.08e3	1.07e3
(200, 100)	N/A	N/A	N/A	N/A	N/A	4.09e3	3.21e4	5.17e3
(200, 200)	N/A	N/A	N/A	N/A	N/A	7.79e3	6.22e4	1.03e4

smaller. We also notice that the number of haplotypes is growing with the length of haplotypes and the number of pedigrees. However, it grows very slowly.

Running time of HANAP

EM-DeCODER is a popular software using the EM algorithm to estimate the haplotype frequencies based on population data. As we have pointed out, it can be used to estimate haplotype frequencies in pedigrees, simply by discarding the pedigree information. Here we also compare the running times of HANAP and EM-DeCODER.

Figure 3 shows their running times over different number of trios (k), length of haplotypes (m) and distributions of allele-probability (P). We can learn from the figure that HANAP runs much quicker than EM-DeCODER, and thus can be applied to much larger instances. We also notice that the running time of both HANAP and EM-DeCODER increase exponentially with the length of haplotypes while increasing near-linearly with the number of trios (the running time of EM-DeCODER is not plotted in Figure 3(b) because haplotypes of length 100 are out of its capability).

Accuracy rate of HANAP

We define a parameter Δ to incarnate the deviation of the estimate haplotype frequencies from the underlying ones. Because the simulation data are generated according to the Θ^*, we recognize that as the underlying true frequencies. Suppose the estimate haplotype set is H^E with frequencies Θ^E. Compare H^E with H^*. Suppose the estimate frequencies of the 20 haplotypes in H^* are θ_1^E, θ_2^E,..., θ_{20}^E. We let,

$$\Delta = \sqrt{\frac{\sum_{i=1}^{20}(\theta_i^E - \theta_i^*)^2}{20}} \qquad (1)$$

Figure 4 shows the deviation of the estimate of HANAP and EM-DeCODER over different number of trios (k), length of haplotypes (m) and distributions of allele-probability (P). We can learn from the figure that the deviation of HANAP is smaller than that of EM-DeCODER, which means HANAP is more accurate. We have also noticed that the deviation of the estimate increases with the length of haplotypes, and decreases with the number of trios.

Table 3: Comparison of number of haplotypes (|*H*|) on a general pedigree

Parameter settings (*m*, *k*)	directly				HANAP			
	P_1	P_2	P_3	P_4	P_1	P_2	P_3	P_4
(20, 20)	2.31e5	1.07e3	3.65e6	1.65e6	20.78	18.34	20.15	20.05
(20, 50)	2.53e5	2.38e3	6.82e6	2.24e6	21.0	19.39	21.2	20.67
(20, 100)	2.96e5	4.51e3	1.08e7	3.38e6	20.9	19.5	20.75	20.8
(200, 20)	N/A	N/A	N/A	N/A	26.8	21.3	22.1	22.4
(200, 50)	N/A	N/A	N/A	N/A	34.4	22.1	23.5	24.2
(200, 100)	N/A	N/A	N/A	N/A	54.7	23.9	28.9	32.8
(500, 20)	N/A	N/A	N/A	N/A	1.41e2	22.4	23.2	23.9
(500, 50)	N/A	N/A	N/A	N/A	2.24e2	29.5	35.7	30.7
(500, 100)	N/A	N/A	N/A	N/A	4.06e2	30.1	41.4	42.5

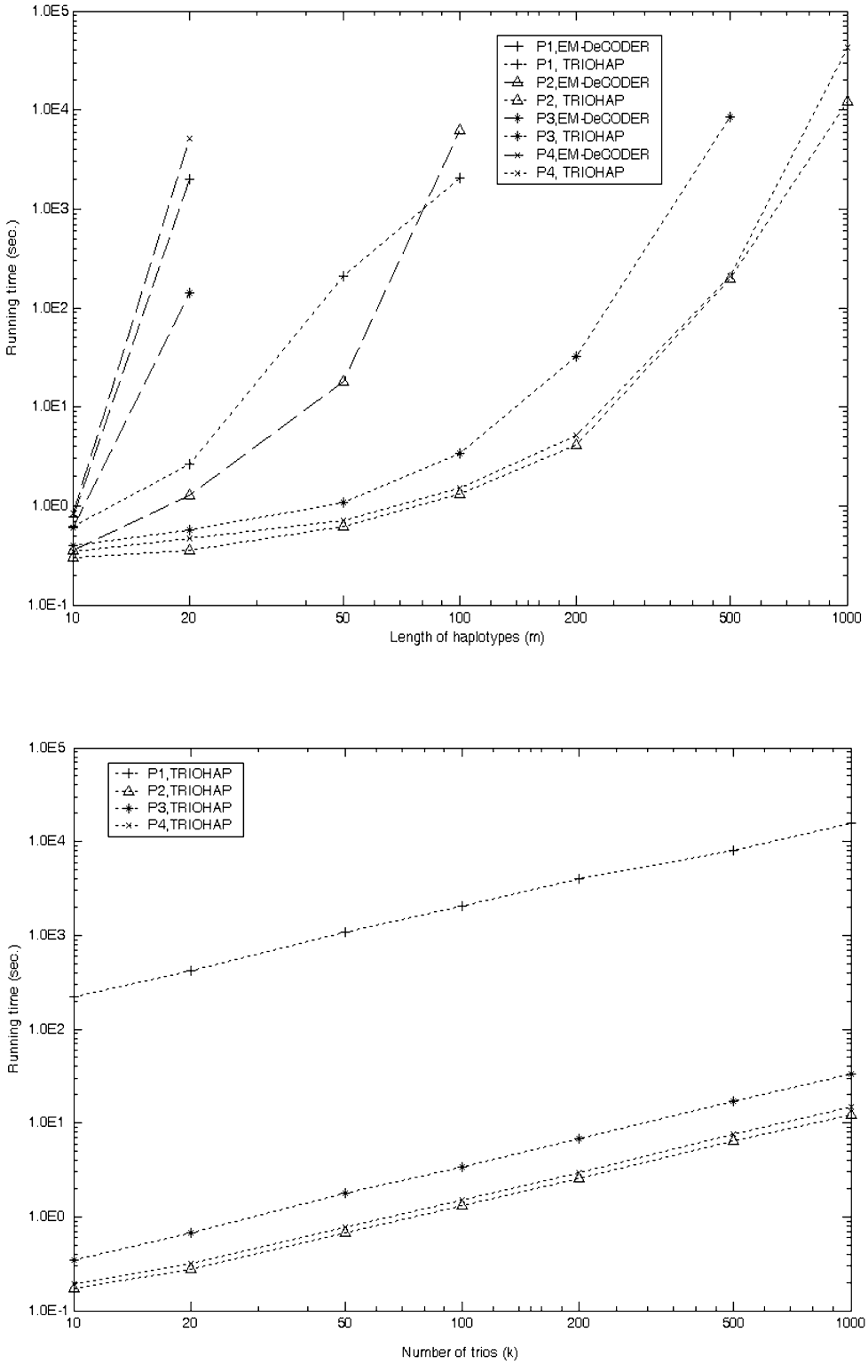

Figure 3
The running time of HANAP and EM-DeCODER: (a) running time *vs*. length of haplotypes (K = 100); (b) running time *vs*. number of trios (m = 100).

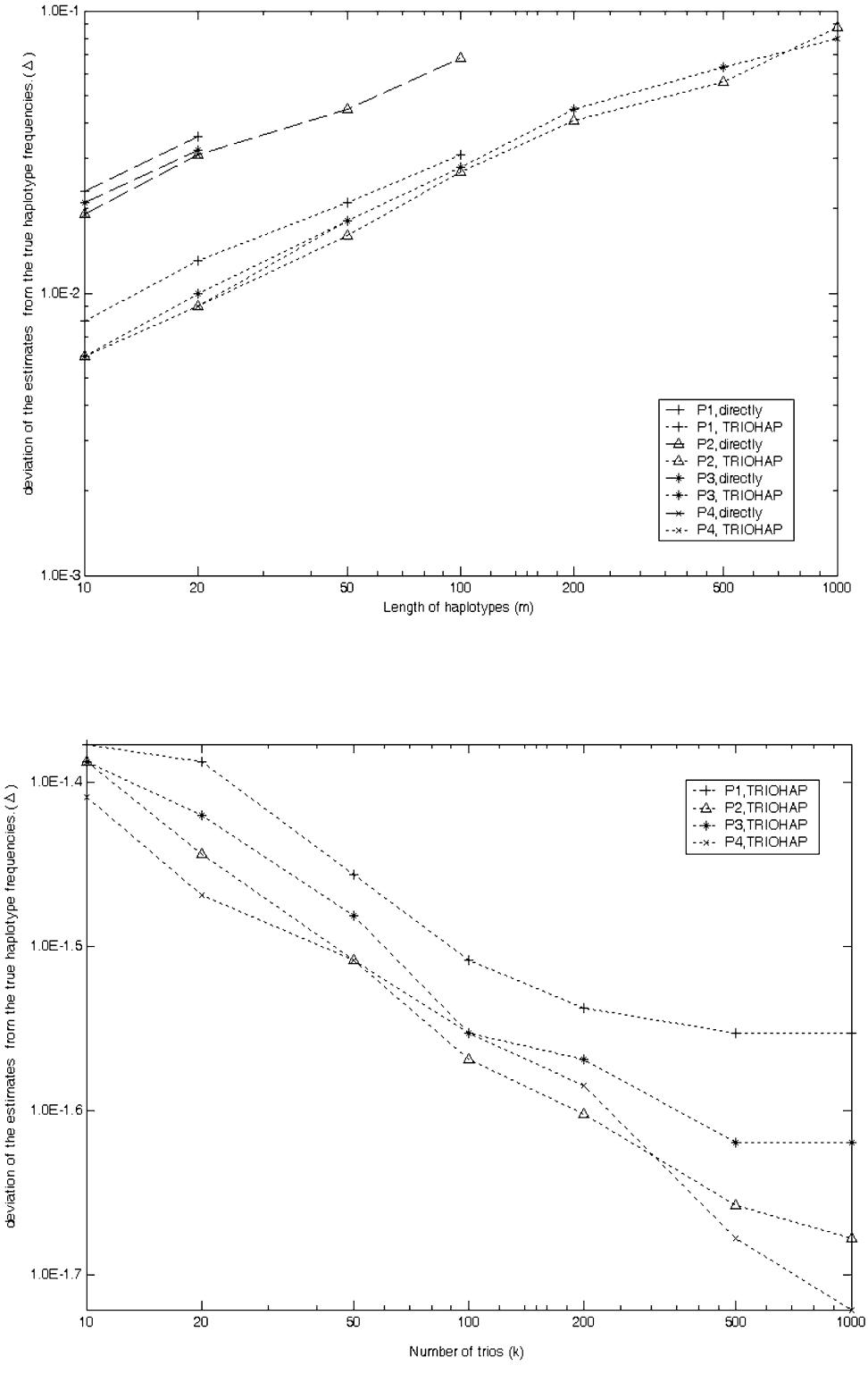

Figure 4
The estimate deviation of HANAP and EM-DeCODER: (a) deviation *vs*. length of haplotypes (K = 100); (b) deviation vs. number of trios (m = 100).

Two real data sets

We also test the efficiency and accuracy of HANAP on two real data sets. The first real data set is from dbMHC|ABDR, a set of 122 trios. Each genotype of these trios contains 31 markers of the same position on chromosome 6, 10 of which are micro-satellite markers and others are SNPs. We run HANAP to find the most frequent haplotypes (with frequencies larger than 0.01). A list of 20 haplotypes is found by HANAP. Their frequencies are shown in Table 4. It only takes HANAP 0.97 second to find these haplotypes while it is out of the capability of EM-DeCODER.

The second data set is from the CEPH database [17], which contains 65 families; each consists of only three generations, usually with four grandparents, two parents and a number of children. Figure 5 in [15] shows a typical family with 21 nodes.

A great portion of the alleles in this data set have not been identified, and will be viewed as missing data. We carefully selected a data set of 28 families (totally 482 nodes) on a block (48 markers) from chromosome 14 (452 markers in total) with no recombination. Both HANAP and PHASE (another widely used software package [3] based on the GS algorithm) are applied to this data set. HANAP inferred 36 haplotypes with frequency larger than 0.01 and PHASE inferred 39 ones, among which 31 are common. Although we are not sure which output is closer to the real cases, the running time of HANAP ($13m24s$) is extremely shorter than that of PHASE ($21h14m$).

Table 4: The frequencies of the 20 most frequent haplotypes found by HANAP

Id	Alleles of Marker Set	θ
1	CCAGGTAGCGCGAAGCATTTCTGTAGTACGA	0.037
2	CCAGGTAGCGCTCTAAGCAACTGGCGACGAG	0.043
3	CCCGGTGGTACGAAGCACAATCGGCGAACAC	0.025
4	CCCGGTGGTACGAAGCACAATCGTAGTACGA	0.102
5	CCCGGTGGTACGAGGTATCTCTAGCAGTCGT	0.016
6	CCCGGTGGTACGAGGTATCTCTAGCGACGAG	0.026
7	CTCGGTGGCGCGAGGTGTATCTATAAACGGC	0.023
8	CTCGGTGGCGCTCTAAGCAGCTGTAGATCGC	0.117
9	CTCGGTGGCGCTCTATGCAACTGGCGACGAG	0.178
10	GACGGTGATATGAGGTATCTCTAGCGTACGC	0.015
11	GACGGTGATATTCTAGGCTTTCGGAAACGAC	0.021
12	GCCGGTCGCGCGAGGCATATCTATAGATCGC	0.027
13	GCCGGTCGCGCTCTAAGCAGCTGTAAACGGC	0.022
14	GTATCGCATATGAAGCACAATCGGAAACGAC	0.078
15	GTATCGCATATGAGGTGTATCTAGAAACGAC	0.041
16	GTATCGCATATGAGGTGTATCTAGCAGTCGT	0.034
17	GTATCGCATATGCGGAGCCGTCAAAATTGGA	0.023
18	GTATCGCATATGCGGCGCCGTCAGAAACGAC	0.055
19	GTATCGCATATTCTAAGCAGCTGTAGATCGC	0.068
20	GTATCGCATATTCTAGGCTTTCGGAAACGAC	0.021
Σ		0.97

Discussion

Complexities of the HCL-linkage analysis haplotyping algorithm

We show that the algorithm runs in $O(mn)$ time and $O(mn)$ space. The pre-process need calculate no more than $3n$ HCL-linkages in no more than n trios. Each HCL-linkage can be computed in $O(m)$ time, so the pre-process can be done in $O(mn)$ time. It takes step 1 $O(n)$ time to construct the rooted tree. In step 2 and step 3, we have to traverse the whole tree, and visit each node for no more than constant times.

When we process the HCL-linkages from the left lowest nuclear family, we should merge the d_1 HCL-linkages at each parent node (if we can), it need $O(d_1m)$ time, here d_1 is the number of children in this family. We need another $O(d_1m)$ time to exchange the HCL-linkage information between the two parents and transfer that to its root R_1 in the search tree **T**. So we need $O(d_1m)$ time in total to process this nuclear family. When we transfer the normalized HCL-linkage set $\Psi_{R_1} = \{\psi_1, \psi_2, ...\psi_k\}$ to the upper nuclear families, we only need to remember that all ψ_i is coming from R_1, so for each ψ_i, it will only take $O(1)$ time to process RE_i, and $O(|LS_i|)$ time to process LS_i and PH_i. The summation time is no more than $O(k + LS_1 + LS_2 + ... + LS_k) = O(m)$ because LS_i are disjoint subsets of $\{1,2,...,m\}$. In other words, the HCL-linkages in one nuclear family won't increase the processing time of its adjacent families. So the total running time to process all nuclear familes is no more than $O(d_1m + d_2m + ... + d_xm) = O(nm)$, here x is the number of nuclear families in the whole pedigree.

We need another $O(mn)$ time to complete step 4. Therefore, the time complexity of this algorithm is $O(mn)$.

For the computation, we need to maintain a data structure to store the HCL-linkage set Ψ_v for each node v; we can maintain the storage always below $O(d_im)$ for nuclear family $\mathbf{F}_i$. So the space complexity of the algorithm is also $O(d_1m + d_2m + ... + d_xm) = O(nm)$.

Effectiveness of the haplotyping phase

Excoffier used the EM algorithm to estimate haplotype frequencies while ignoring the pedigree information. Here we adopt a two-stage method, which tries to reduce the number of possible haplotypes to be considered in the stage of estimation by utilizing the relatives' information to do haplotyping at first.

Suppose we are estimating haplotype frequencies in trios and each haplotype consists of m biallelic SNP loci. For locus i, suppose that i happens to be one state with a probability of p_i, and to be the other state with a probability of $(1 - p_i)$. Then locus i of the genotype is heterozygous with the probability of $2p_i(1 - p_i)$. Suppose the expected value of p_i is p, then the genotype is expected to have $2p(1 - p) \cdot m$ heterozygous loci. As a consequence, a total number of $2^{2p(1-p)\cdot m-1}$ possible haplotype pairs is expected to be considered if we use the EM algorithm directly. However, the probability that locus i in a trio is ambiguous is $2p_i(1 - p_i) \cdot 2p_i(1 - p_i) \cdot 2/4 = 2p_i^2(1 - p_i)^2$. So the expected number of possible haplotype configurations for the trio is $2^{2p^2(1-p)^2\cdot m}$. If $p = 1/2$, our method can handle $\lambda = (2p(1 - p))/(2p^2(1 - p)^2) = 4$ times longer genotypes than Excoffier's methods. Moreover, in most cases, the more frequent allele at one locus appears with a probability of more than 0.9, so our method usually can handle $\lambda = 1/p(1 - p) > 10$ times longer genotypes.

Furthermore, if each locus of the haplotype is a micro-satellite locus, and it has l different alleles: $a_1, a_2,...,a_l$, each appears with the probability of $p_1, p_2,...,p_l$. then the expected number of possible haplotype pairs for a genotype is $2^{\sum_{i\neq j} p_i p_j \cdot m-1}$, the expected number of feasible haplotype configurations for a trio is $2^{\sum_{i\neq j} p_i^2 p_j^2 \cdot m}$, so our method usually can handle $\lambda = \sum_{i\neq j} p_i p_j \Big/ \sum_{i\neq j} p_i^2 p_j^2$ times longer genotypes. For example, when $l = 8$, and $p_1 = p_2 = ... = p_8 = 1/8$, $\lambda = 64$, *i.e.* our method can be applied to cases of much larger scale.

Conclusion

We present a two-stage method to do haplotyping and to estimate haplotype frequencies for pedigree genotype data in this paper. Given a set of pedigrees, it firstly determines all feasible haplotype configurations for each pedigree, then uses the EM algorithm to estimate the haplotype frequencies based on the inferred haplotype configurations. Because a large number of illegal haplotypes have been eliminated from the possible haplotype list, our method is both more efficient and more accurate. The experimental results show that, HANAP runs much faster than EM-DeCODER, and thus can be applied to much larger scale of instances. Moreover, the deviation of the estimate of HANAP is smaller than that of EM-DeCODER, which means it is more accurate.

Our method suggests that pedigree information is of great importance in haplotype analysis. It can be used to speedup estimation process, and to improve estimation accuracy as well. The result also demonstrates that whole haplotype configuration space can be substitute by the space of zero-recombinant haplotype configurations in haplotype frequency estimation, especially when the considered haplotype block is relatively short.

Authors' contributions

QZ proposed the whole estimation framework and both the haplotyping and the estimation algorithms, and wrote the manuscript. YZ implemented the software and helped writing the manuscript. GC and YX participated to the design of the study and wrote the manuscript. All authors read and approved the final manuscript.

Appendix

Correctness of the HCL-linkage analysis haplotyping algorithm

First, we point out that every consistent haplotype configuration for pedigree P should be consistent with all HCL-linkages calculated by the unique partial solutions within trios, and our merge and transfer operations keep this feature during the whole process, *i.e.* if ζ is a haplotype configuration for P, and ζ is consistent with all the HCL-linkages in the pedigree, it should also be consistent with those after the merge and transfer operations.

Obviously, HCL-linkages are sufficient to generate a consistent haplotype configuration within a trio. We prove that:

Lemma

If it exists consistent haplotype configuration for a nuclear family (F, M, C_1, C_2,...,C_d), *every imputing schema s of arbitrarily imputing the* (*compound*) *alleles at one node can output one feasible solution* ζ. *Contrarily, if one imputing schema ends with incompatibleness, there is no consistent haplotype configuration*.

Proof

Firstly, HCL-linkages are necessary for constructing the consistent haplotype configurations, which means that if there is a feasible solution, it should correspond to one imputing schema.

Secondly, we show that if one imputing schema outputs a feasible solution, all the schemas output feasible solutions.

In particular, for a trio (F, M, C_1), without loss of generality, we suppose that Step 4 starts by imputing the "alleles" of node F. For a specified "locus" i, the two alleles of F, M and C_1 are denoted as (a, b), (c, d) and (e, f), and the par-

tial haplotypes from locus 1 to locus (i - 1) are denoted as (ph_1, ph_2), (ph_3, ph_4) and (ph_5, ph_6).

Suppose in schema s, allele a is imputed to ph_1, denoted as $ph_1 \leftarrow a$, and $ph_2 \leftarrow b$, $ph_3 \leftarrow c$, $ph_4 \leftarrow d$, $ph_5 \leftarrow e$, $ph_6 \leftarrow f$. If s outputs a consistent haplotype configuration for trio (F, M, C), we proof that s': $ph_2 \leftarrow a$, $ph_1 \leftarrow b$, $ph_4 \leftarrow c$, $ph_3 \leftarrow d$, $ph_6 \leftarrow e$, $ph_5 \leftarrow f$ outputs another consistent haplotype configuration.

The loci from 1 to (i - 1) can be viewed as a compound locus I. Let's refer to Table 1, because we can impute a or b to ph_1, either or both of "locus" I and i should be *ambiguous*. Else, they will be linked to a bigger compound "locus" in Step 2 or 3 by HCL-linkages. Whatever a or b will be linked with ph_1, we can not impute that arbitrarily in Step 4. Without loss of generality, we assume that locus i is ambiguous, which means that $a \neq b$, and $\{a, b\} = \{c, d\} = \{e, f\}$ (please refer to Table 1). We can prove that s' is also a consistent haplotype configuration by enumeration. Because nuclear family (F, M, C_1, C_2,...,C_d) is the intersection of trio (F, M, C_1), (F, M, C_2),...,(F, M, C_d), the above prove shows that both both s: $ph_1 \leftarrow a$, $ph_2 \leftarrow b$, $ph_3 \leftarrow c$, $ph_4 \leftarrow d$ and s': $ph_2 \leftarrow a$, $ph_1 \leftarrow b$, $ph_4 \leftarrow c$, $ph_3 \leftarrow d$ will lead to a consistent haplotype configuration for the family.

We call s to s' a walk step by switching $ph_1 \leftarrow a$ to $ph_1 \leftarrow b$. Obviously, for any two imputing schemata s_1 and s_2, we can transfer s_1 to s_2 by consecutive walk steps. So $s_1$and s_2 will lead to a consistent haplotype configuration or neither can.

This lemma indicates that our algorithm works in a nuclear family. We now can prove the correctness of our HCL-linkage analysis haplotyping algorithm by induction, *i.e.* if there is at least one feasible solution for a general pedigree, HCL-linkages are sufficient to generate all solutions.

Suppose that the root R has multiple child mating nodes: O_1,...,O_r, each represents a nuclear family; and the algorithm works in all sub-trees, *i.e.* all of the feasible solutions for those sub-trees can be directly deduced from the HCL-linkages collected from the sub-trees and stored at their roots. From the former lemma, we know that if incompatibleness of type II occurs, there is no feasible solution. We assume that there is no incompatibleness of type II and there are always feasible solutions for all sub-trees. Suppose that haplotype configuration ζ is consistent with all the HCL-linkages at root R (and all the HCL-linkages in the P consequently). Then ζ should be consistent haplotype configuration when restricted to any nuclear family of O_1,...,O_r, and it should also be consistent with the HCL-linkages at the root of lower sub-trees of O_1,...,O_r. By induction, ζ should be a feasible solution when restricted to any of these sub-trees. So ζ is a consistent haplotype configuration of the whole pedigree P.

Acknowledgements

This work is supported by the National Science Foundation of China under the grant No.60533020.

This article has been published as part of *BMC Bioinformatics* Volume 7, Supplement 4, 2006: Symposium of Computations in Bioinformatics and Bioscience (SCBB06). The full contents of the supplement are available online at http://www.biomedcentral.com/1471-2105/7?issue=S4.

References

1. Excoffier L, Slatkin M: **Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population.** *Mol Biol Evol* 1995, **12:**921-927.
2. Hawley ME, Kidd KK: **HAPLO: a program using the EM algorithm to estimate the frequencies of multi-site haplotypes.** *J Hered* 1995, **86(5):**409-11.
3. Stephens M, Smith NJ, Donnelly P: **A new statistical method for haplotype reconstruction for population data.** *Am J Hum Genet* 2001, **68:**978-989.
4. Niu T, Qin ZS, Xu X, Liu JS: **Bayesian haplotype inference for multiple linked single nucleotide polymorphisms.** *Am J Hum Genet* 2002, **70:**157-169.
5. Griffiths A, Gelbart W, Lewontin R, Miller J: **Modern Genetic Analysis: Integrating Genes and Genomes.** *New York: W.H. Freeman and Company*; 2002.
6. Cox R, Bouzekri N, *et al.*: **AngiotensinIconverting enzyme (ACE) plasma concentration is influenced by multiple ACE-linked quantitative trait nucleotides.** *Hum Mol Genet* 2002, **11:**2969-2977.
7. Qian D, Beckmann L: **Minimum recombinant haplotyping in pedigrees.** *Am J Hum Genet* 2002, **70(6):**1434-1445.
8. Li J, Jiang T: **Efficient rule-based haplotyping algorithms for pedigree data.** *Proc of RECOMB* 2003:197-206.
9. Li J, Jiang T: **Efficient inference of haplotypes from genotypes on a pedigree.** *J Bioinfo Comp Biol* 2003, **1(1):**41-69.
10. Wijsman EM: **A deductive method of haplotype analysis in pedigrees.** *Am J Hum Genet* 1987, **41(3):**356-373.
11. O'Connell JR: **Zero-recombinant haplotyping: applications to fine mapping using SNPs.** *Genet Epidemiol* 2000, **19(Suppl 1):**S64-70.
12. Elston RC, Stewart J: **A general model for the genetic analysis of pedigree data.** *Human Heredity* 1971, **21:**523-542.
13. Fallin D, Schork NJ: **Accuracy of haplotype frequency estimation for biallelic loci, via the expectation maximization algorithm for unphased diploid genotype data.** *Am J Hum Genet* 2000, **67:**947-959.
14. Zhao H, *et al.*: **Transmission/disequilibrium tests using multiple tightly linked markers.** *Am J Hum Genet* 2000, **67:**936-946.
15. Zhang Q, Chin FYL, Shen H: **Minimum Parent-Offspring Recombination Haplotype Inference in Pedigrees.** *Transactions on Computational Systems Biology LNBI 3680-0100* 2005, **2:**100-12.
16. Zhang Q, Che H, Zhou Z, Chen G: **Comparative study on different approaches to in silico haplotyping.** In *Technical report Dept of Computer Science and Technology, University of Science and Technology of China*; 2003.
17. **The CEPH genotype database** [http://www.cephb.fr/]