Efficient Inference of Haplotypes from Genotypes on a Pedigree with Mutations and Missing Alleles (Extented Abstract)

Wei-Bung Wang and Tao Jiang

Department of Computer Science, University of California, Riverside, CA 92506, USA
weiw@cs.ucr.edu, jiang@cs.ucr.edu

Abstract. Driven by the international HapMap project, the haplotype inference problem has become an important topic in the computational biology community. In this paper, we study how to efficiently infer haplotypes from genotypes of related individuals as given by a pedigree. Our assumption is that the input pedigree data may contain *de novo* mutations and missing alleles but is free of genotyping errors and recombinants, which is usually true for tightly linked markers. We formulate the problem as a combinatorial optimization problem, called the *minimum mutation haplotype configuration* (MMHC) problem, where we seek haplotypes consistent with the given genotypes that incur no recombinants and require the minimum number of mutations. This extends the well studied *zero-recombinant haplotype configuration* (ZRHC) problem. Although ZRHC is polynomial-time solvable, MMHC is NP-hard. We construct an *integer linear program* (ILP) for MMHC using the system of linear equations over the field $F(2)$ that has been developed recently to solve ZRHC. Since the number of constraints in the ILP is large (exponentially large in the general case), we present an incremental approach for solving the ILP where we gradually add the constraints to a standard ILP solver until a feasible haplotype configuration is found. Our preliminary experiments on simulated data demonstrate that the method is very efficient on large pedigrees and can infer haplotypes very accurately as well as recover most of the mutations and missing alleles correctly.

1 Introduction

Human beings have been fighting against diseases such as cancer, stroke, heart disease, asthma, depression, and schizophrenia for decades. It is believed that many of these diseases are caused by genetic factors. *Gene mapping*, which attempts to establish connections between diseases and some specific genetic variations, is a very important and active area of genetics. More specifically, it aims at locating genes of interest (*e.g.* genes responsible for certain diseases) relative to genetic markers (such as microsatellites and *single nucleotide polymorphisms*, or SNPs) on chromosomes. A set of genetic markers and their positions (called *marker loci*) define a *genetic map* of chromosomes. In *diploid* organisms like human, chromosomes (other than sex chromosomes) form pairs. Each pair of chromosomes consists of a *paternal* chromosome inherited from the father and a *maternal* chromosome inherited from the mother. Hence, each genetic marker on a pair of chromosomes occurs at the same location of both paternal and maternal chromosomes. However, the marker may have different states (called *alleles*) on the two chromosomes. The set of its two alleles is called the *genotype* of the marker and the assignment of the two alleles to the paternal and maternal chromosomes is called the *haplotype* (or *phase*) of the marker. The haplotype information of genetic markers is of tremendous value to gene mapping and other genetic analyses (such as linkage analysis) because it gives a more accurate description of the inheritance process than the genotype information. Its importance can also be seen from the international HapMap project launched in 2002 [19]. Since genotype data instead of haplotype data are routinely collected in practice, especially in large-scale sequencing projects, due to cost considerations, efficient and accurate computational methods for the inference of haplotypes from genotypes over a set of marker loci, which is also commonly referred to as *phasing*, have been extensively studied in the literature. See [11] for a recent survey on these methods as well as the basic concepts involved in haplotype inference.

The existing computational methods for haplotype inference can be divided into three groups according to the genotype data that they deal with: methods for *population* data involving unrelated individuals (see *e.g.* [6,13,17]), methods for *pedigree* data consisting of individuals (typically from an extended family) that are related by the parent-child relationship (see *e.g.* [1,8,9,10,15,16,21,23]), and methods for *pooled samples* (see *e.g.* [20,22]). The methods for population data usually consider tightly linked markers that may involve mutations but no recombinants, while the methods for pedigree data usually assume that the data may have zero or few recombinants but is free of mutations (*i.e.* the Mendelian law of inheritance holds). Here, we are interested in only pedigree data.

Some real pedigree data may actually contain mutations. In particular, a *de novo mutation* is a mutation that is present for the first time in a family member as a result of a mutation in a germ cell (egg or sperm) of one of the parents or in the fertilized egg itself. It has been found that the detection and analysis of mutations in a pedigree could provide a good alternative for some genetic variation research [3,5,14]. In fact, Ellegren [5] has stated that "To reveal the mutational contribution to overall genetic variability, the most straightforward and conclusive way is the direct detection of mutation events in pedigree genotyping." However, *de novo* mutations violate the Mendelian law of inheritance, and hence pedigree data with such mutations cannot be properly handled by the above common haplotype inference methods. When these methods are faced with data with mutations, they typically treat the loci involving mutations as genotyping errors and delete such loci. Very few haplotype inference methods in the literature deal with pedigree data that contain mutations (one such method is a genetic algorithm in [18]).

In this paper, we study haplotype inference on pedigree data on tightly linked markers that have no recombinants but may contain a small number of *de novo* mutations (or simply, mutations). Since mutation is a rare event, we formulate the problem as a combinatorial optimization problem, called the *minimum mutation haplotype configuration* (MMHC) problem, where we look for a haplotype solution consistent with the given genotype data that incur no recombinants and require the minimum number of mutations. Our hypothesis is a solution with the minimum number of mutations is likely the true solution. Moreover, we are only interested in solutions where each locus has at most one mutation in the pedigree. This restriction is reasonable given Kimura's infinite-site model *et al.* [7] which suggests that the probability of multiple mutations at the same locus is low enough to be negligible. This extends the well studied *zero-recombinant haplotype configuration* (ZRHC) problem where we try to find a consistent haplotype solution incurring no recombinants or mutations. Although ZRHC is polynomial-time solvable [9], we can prove that MMHC is NP-hard by a reduction from NAE-3SAT (the proof is omitted in this extended abstract). We construct an *integer linear program* (ILP) for MMHC using the system of linear equations over the field $F(2)$ that has been developed in [9,12,21] for solving ZRHC in almost linear time. Since the number of constraints in the ILP is quite large (exponentially large in general) when the input pedigree is large, we present an incremental approach for solving the ILP.

An outline of our incremental approach is as follows. Given a pedigree data, we set up a system of linear equations over $F(2)$ introduced in [12,21] for ZRHC, but conditional on mutations. We convert the linear system to an ILP instance for MMHC where the constraints generally describe the relation between the equations and mutations. A small set of the constraints in the ILP are identified as the *core* constraints, and a standard ILP solver GLPK (the GNU Linear Programming Kit from `http://www.gnu.org/softward/glpk`) is invoked on the partial ILP instance with only the core constraints. The ILP solution describes an assignment of mutations in the pedigree which can be used to remove the conditions in the linear system. By using Gaussian elimination, we can check if the linear system is consistent. If it is consistent, a haplotype configuration (with the minimum number of mutations) is returned. Otherwise, we find the inconsistent equations and add some new constraints to the core to force their consistency. This process is repeated until an ILP solution that satisfies its corresponding linear system has been found. Note that, the incremental approach to solving the ILP is crucial here because the ILP instance cannot be efficiently and explicitly constructed as its number of constraints grows exponentially in the pedigree size in general. Also note that, with the advance in sequencing technology, larger and larger pedigrees are being genotyped and analyzed in practice. For example, in [2,4], haplotype inference was performed on pedigrees of sizes 368 and 1149, respectively.

We have implemented the algorithm and tested it on pedigree data that were simulated with random mutations and missing alleles. (Real pedigree data often have up to 20% missing alleles.) The experimental results demonstrate that our method can infer haplotypes with a very high accuracy. It can also detect most of the mutations and impute most of the missing alleles correctly. Moreover, it is found that the algorithm usually terminates after a small number of iterations without ever having to invoking ILP solver on the complete ILP instance consisting of all the constraints. As a comparison, we have also considered the straightforward approach for solving the ILP with all the constraints considered at once on binary tree pedigrees (*i.e.* each pair of parents has only one child). The ILP instance can be efficiently constructed for binary tree pedigrees. It is found that our algorithm is much faster than the straightforward approach.

The rest of the paper is organized as follows. In Section 2, we incorporate mutations into the system of linear equations introduced in [12,21] for ZRHC to obtain a system of conditional linear equations for MMHC. Section 3 describes the ILP formulation for MMHC, and the incremental approach for solving the ILP. In Section 4, we discuss the implementation of the algorithm and test its performance on some simulated pedigree data with random mutations and missing alleles. Section 5 concludes the paper with a few remarks.

2 A System of Conditional Linear Equations for MMHC

We review the system of linear equations over $F(2)$ introduced in [12,21] for solving ZRHC and extend the system to take into account mutations.

2.1 The Linear System

Let n denote the number of the individuals in the input pedigree and m the number of marker loci of each individual. For simplicity, we assume in this paper that all alleles are bi-allelic (denoted as 0 or 1) and the input pedigree is free of mating loops (and thus a tree pedigrees). Tree pedigrees are very common among human pedigrees. Our techniques can be extended to general pedigrees. The genotype of individual j is denoted as a ternary vector $\mathbf{g}_j$ whose kth entry $g_j[k]$ represents the genotype at locus k of individual j as follows:

$$\begin{cases} g_j[k] = 0 & \text{if both alleles are 0's} \\ g_j[k] = 1 & \text{if both alleles are 1's} \\ g_j[k] = 2 & \text{if the locus is heterozygous} \end{cases} \tag{1}$$

The value of $g_j[k]$ is *unknown* if the alleles are missing. For each locus k of individual j, we define a binary variable $p_j[k]$ over $F(2)$ to indicate the paternal allele at the locus:

$$\begin{cases} p_j[k] = 0 & \text{if } g_j[k] = 0 \\ p_j[k] = 1 & \text{if } g_j[k] = 1 \\ p_j[k] = 0 & \text{if } g_j[k] = 2 \text{ and allele 0 is paternal} \\ p_j[k] = 1 & \text{if } g_j[k] = 2 \text{ and allele 1 is paternal} \end{cases} \tag{2}$$

In other words, the binary vector $\mathbf{p}_j$ represents the paternal haplotype of individual j. To represent the maternal haplotype, we need another binary vector $\mathbf{w}_j$ to indicate if each locus of individual j is heterozygous. That is, $w_j[k] = 0$ if

$g_j[k] = 0$ or 1, and $w_j[k] = 1$ if $g_j[k] = 2$. Clearly, the sum $\mathbf{p}_j + \mathbf{w}_j$ (over $F(2)$) represents the maternal haplotype of individual j.

Suppose that individual i is a parent of individual j. To unify the representation of the haplotype that j inherited from i, define a binary vector $\mathbf{d}_{i,j}$ as follows: $\mathbf{d}_{i,j} = 0$ if i is j's father and $\mathbf{d}_{i,j} = \mathbf{w}_j$ if i is j's mother. Therefore, $\mathbf{p}_j + \mathbf{d}_{i,j}$ represents the haplotype that j got from i. Define $h_{i,j} = 0$ if $\mathbf{p}_j + \mathbf{d}_{i,j}$ is i's paternal haplotype and $h_{i,j} = 1$ otherwise. Then $\mathbf{p}_i + h_{i,j} \cdot \mathbf{w}_i$ represents the haplotype that i passed to j. The binary variables $h_{i,j}$ thus fully describe the inheritance pattern in an ZRHC instance. Finally, define $\mu_{i,j}[k] = 1$ if the there is a mutation at locus k when i passes the haplotype $\mathbf{p}_i + h_{i,j} \cdot \mathbf{w}_i$ to j, and $\mu_{i,j}[k] = 0$ otherwise. For technical reasons, we view $\mu_{i,j}[k]$ as an integer from $\mathbb{Z}$ instead of $F(2)$. For convenience, we make these three vectors symmetric by defining $\mathbf{d}_{j,i} = \mathbf{d}_{i,j}$, $h_{j,i} = h_{i,j}$, and $\mu_{j,i} = \mu_{i,j}$. Using these notations, we can derive a *conditional equation* over $F(2)$:

$$\begin{cases} p_i[k] + h_{i,j} \cdot w_i[k] = p_j[k] + d_{i,j}[k] & \text{if } \mu_{i,j}[k] = 0 \\ p_i[k] + h_{i,j} \cdot w_i[k] = p_j[k] + d_{i,j}[k] + 1 & \text{if } \mu_{i,j}[k] = 1 \end{cases} \tag{3}$$

Since we assume that each locus has at most one mutation in the pedigree,

$$0 \leq \sum_{i,j} \mu_{i,j}[k] \leq 1 \quad \forall k \tag{4}$$

Note that the summation is over $\mathbb{Z}$ instead of $F(2)$. Hence, the MMHC problem can be formally defined as follows. Given an input pedigree and genotype data $\mathbf{g}_j$ for each individual j, find a solution to each $\mathbf{p}_j$, $h_{i,j}$ and $\mu_{i,j}$ that satisfies all the (conditional) constraints in Equations (3) and (4) and minimizes the sum $\sum_{i,j,k} \mu_{i,j}[k]$.

2.2 Pre-Determined Variables

The above linear system has $O(mn)$ variables and equations. As in [12,21], we can convert the system to an equivalent linear system involving only the h-variables which is much smaller (there are only $O(n)$ h-variables). This requires us to *pre-determine* the values of some p-variables. The situation is complicated a little bit by the presence of the μ-variables.

Let us consider a p-variable $p_j[k]$ where the marker of individual j at locus k is not missing, and several scenarios.

1. $g_j[k] \neq 2$. By Equation (2), $p_j[k] = g_j[k]$. In this case, $p_j[k]$ is *pre-determined.* We will refer to $p_j[k]$ as the *intended p-value* of the locus, denoted as $v(j,k) = p_j[k]$.
2. $g_j[k] = 2$ and exactly one parent, denoted as i, is homozygous at locus k. See Figure 1(a). We have $w_i[k] = 0$ by definition. According to Equation (3), $p_j[k]$ is known if and only if $\mu_{i,j}[k]$ is known. We say that $p_j[k]$ is *semi-determined* in this case. We also define $\mu_{i,j}[k]$ as the *anchor* of $p_j[k]$ and denote $a(j,k) = \{\mu_{i,j}[k]\}$. Since the value of $p_j[k]$ on the condition $\mu_{i,j}[k] = 0$ is preferred, we denote $v(j,k) = g_i[k] + d_{i,j}[k]$.

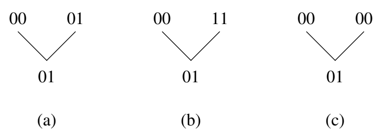

Fig. 1. Determining a p-variable. Consider the p-value of the child in the trio. (a) It equals 0 as long as there is no mutation from the father and it is semi-determined. (b) It equals 0 and there cannot be any mutation. It is pre-determined. (c) It is undetermined but there must be a mutation. It is doubly-determined.

3. $g_j[k] = 2$, both parents i_1 and i_2 of j are homozygous at locus k, and $g_{i_1}[k] \neq g_{i_2}[k]$. See Figure 1(b). Since each locus has at most one mutation, $\mu_{i_1,j}[k]$ and $\mu_{i_2,j}[k]$ cannot both be 1. Hence, $\mu_{i_1,j}[k] = \mu_{i_2,j}[k] = 0$. In this case, $p_j[k]$ is pre-determined, and we denote $v(j,k) = p_{i_1}[k] + d_{i_1,j}[k]$.
4. $g_j[k] = 2$, both parents i_1 and i_2 are homozygous at locus k, and $g_{i_1}[k] = g_{i_2}[k]$. See Figure 1(c). In this case, one of $\mu_{i_1,j}[k]$ and $\mu_{i_2,j}[k]$ equals 1 and the other 0. Thus, $p_j[k]$ has two anchors: $a_1(j,k) = \{\mu_{i_1,j}[k]\}$ and $a_2(j,k) = \{\mu_{i_2,j}[k]\}$. Each anchor gives rise to a preferred value for $p_j[k]$, $v_1(j,k) = p_{i_1}[k] + d_{i_1,j}[k]$ and $v_2(j,k) = p_{i_2}[k] + d_{i_2,j}[k]$, respectively. In this case we call $p_j[k]$ *doubly-determined*.
5. All other cases. The variable $p_j[k]$ is *undetermined* and the variable $v(j,k)$ is undefined.

If a $p_j[k]$ is pre-determined or undetermined, we define $a(j,k) = \emptyset$. Similarly, we might be able to pre-determine μ-variable $\mu_{i,j}[k]$ in some cases.

1. $g_i[k] = g_j[k] \neq 2$. Since the top equation in Equation (3) holds, we let $\mu_{i,j}[k] = 0$ and it is pre-determined.
2. $g_i[k] \neq g_j[k]$ and both loci are homozygous. Since the bottom equation in Equation (3) holds, we set $\mu_{i,j}[k] = 1$. This μ-variable is pre-determined. Moreover, all the other μ-variables at locus k must equal 0 and are pre-determined too.
3. Some p-variable at locus k is doubly determined. All the μ-variables at locus k other than this p-variable's anchors must equal 0 and are thus pre-determined.
4. All other cases. The variable $\mu_{i,j}[k]$ stays undetermined.

2.3 A More Compact Linear System

Following [12,21], we can set up a linear system in terms of the h-variables. The idea is to consider paths in the pedigree connecting individuals with pre/semi/ doubly-determined p-variables and derive (conditional) equality constraints on the h-variables on such paths based on Equation (3).

Consider a locus k and a path $j_0, j_1, \ldots, j_r$ in the input (tree) pedigree, where individuals j_i and j_{i+1} have the parent-child relationship. Suppose that $p_{j_0}[k]$ and $p_{j_r}[k]$ are pre-determined, semi-determined or doubly-determined, and

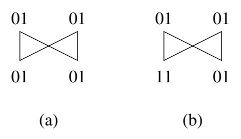

Fig. 2. Two possible cycle constraints from a local cycle. (a) The sum of the four h-variables is 0. (b) The sum of the four h-variables is 1.

$g_{j_1}[k] = \cdots = g_{j_{r-1}}[k] = 2$. We call the path $j_0, j_1, \ldots, j_r$ an *all-heterozygous path* at locus k. If $p_{j_0}[k]$ and $p_{j_r}[k]$ are pre-determined or semi-determined, we define a *path constraint* connecting j_0 and j_r:

$$v(j_0, k) + v(j_r, k) + \sum_{i=0}^{r-1} \left(h_{j_i, j_{i+1}} + d_{j_i, j_{i+1}}[k]\right) = 0$$
$$\text{if all elements in } a(j_0, k) \cup a(j_r, k) \cup \bigcup_{i=0}^{r-1} \{\mu_{j_i, j_{i+1}}[k]\} \text{ equal } 0 \qquad (5)$$

If we denote $\mathcal{M} = a(j_0, k) \cup a(j_r, k) \cup \bigcup_{i=0}^{r-1}\{\mu_{j_i,j_{i+1}}[k]\}$, $\mathcal{H} = \bigcup_{i=0}^{r-1}\{h_{j_i,j_{i+1}}\}$, and $c = v(j_0, k) + v(j_r, k) + \sum_{i=0}^{r-1} d_{j_i,j_{i+1}}[k]$, then the path constraint can also be represented by the triple $(\mathcal{H}, \mathcal{M}, c)$ which denotes:

$$\sum_{h_{i,j} \in \mathcal{H}} h_{i,j} = c \qquad \text{iff } \mu_{i,j}[k] = 0 \quad \forall \mu_{i,j}[k] \in \mathcal{M} \qquad (6)$$

If j_0 or j_r is doubly-determined, we can construct two path constraints in the same way: one using $v_1(\cdot)$ and $a_1(\cdot)$ and the other using $v_2(\cdot)$ and $a_2(\cdot)$.

Consider a *local cycle* consisting of father i_1, mother i_2, and two adjacent children j_1, j_2. If both parents are heterozygous at locus k, we can obtain four conditional equations from Equation (3) by replacing i with i_1, i_2, and j with j_1, j_2. (See Figure 2.) The summation of these conditional equations forms a *cycle constraint*:

$$\begin{aligned} h_{i_1,j_1} + h_{i_1,j_2} + h_{i_2,j_1} + h_{i_2,j_2} &= d_{i_1,j_1}[k] + d_{i_1,j_2}[k] + d_{i_2,j_1}[k] + d_{i_2,j_2}[k] \\ &= w_{j_1}[k] + w_{j_2}[k] \\ \text{iff } \mu_{i_1,j_1}[k] &= \mu_{i_1,j_2}[k] = \mu_{i_2,j_1}[k] = \mu_{i_2,j_2}[k] = 0 \end{aligned} \qquad (7)$$

This constraint will also be denoted as $(\mathcal{H}, \mathcal{M}, c)$ where $\mathcal{H} = \{h_{i_1,j_1}, h_{i_1,j_2}, h_{i_2,j_1}, h_{i_2,j_2}\}$, $\mathcal{M} = \{\mu_{i_1,j_1}[k], \mu_{i_1,j_2}[k], \mu_{i_2,j_1}[k], \mu_{i_2,j_2}[k]\}$, and $c = w_{j_1}[k] + w_{j_2}[k]$.

If both parents are homozygous at locus k, then the p-variables of both children must be pre-determined or doubly-determined. However, the two children are not connected by any all-heterozygous path and thus no path constraint is derived. On the other hand, if exactly one parent is heterozygous at locus k, then both children are semi-determined and there is a path constraint between the two children through the heterozygous parent.

For each locus and every pair of pre/semi/doubly-determined p-variables connected by an all-heterozygous path, we construct a path constraint (or two if one of the p-variables is doubly-determined, or four if both p-variables are doubly-determined) as above. Since the pedigree is a tree, the number of such path constraints is at most $O(mn)$. Similarly, for each locus and local cycle, if both parents are heterozygous at the locus, we construct a cycle constraint as above. The number of such cycle constraints is also bounded by $O(mn)$. Let $\mathcal{E}$ denote the set of these constraints.

The results in [12] show that the linear system formed by the above constraints (without the conditions) in terms of the h-variables is equivalent to the linear system defined by Equation (3) (without the conditions) in terms of the h- and p-variables. In other words, a feasible solution to the h-variable can be extended to a feasible solution to both the h- and p-variables. It is easy to see that the same equivalence holds with the conditions.

Note that, loci with missing alleles could be included in the linear system in Equation (3) (as p-variables). However, they are excluded from the above path/cycle constraints on h-variables. Some of the missing alleles will be imputed using Equation (3) after the h-variables are determined.

3 The ILP for MMHC and Incremental Approach

We construct an ILP for MMHC based on the above linear system in h-variables. Recall that the objective of the ILP is

$$\text{Minimize} \quad \sum_{i,j,k} \mu_{i,j}[k]. \tag{8}$$

We give all the constraints of the ILP in Sections 3.1 and 3.2. Section 3.3 presents more details of the incremental approach to solving the ILP. In Section 3.4, we describe how to obtain a solution for MMHC after solving the ILP (and the linear system) and deal with missing alleles.

3.1 The Core Constraints

All the constraints in Equation (4) are core constraints of the ILP. For each path/cycle constraint $(\mathcal{H}, \mathcal{M}, c)$ in $\mathcal{E}$, we introduce an *equation variable*:

$$E_{\mathcal{H}} = \sum_{h_{i,j} \in \mathcal{H}} h_{i,j} \tag{9}$$

We then add an *equation constraint* for each $(\mathcal{H}, \mathcal{M}, c)$:

$$\begin{cases} E_{\mathcal{H}} - \sum_{\mu_{i,j}[k] \in \mathcal{M}} \mu_{i,j}[k] = 0 & \text{if } c = 0 \\ E_{\mathcal{H}} + \sum_{\mu_{i,j}[k] \in \mathcal{M}} \mu_{i,j}[k] = 1 & \text{if } c = 1 \end{cases} \tag{10}$$

In other words, either the linear equation in $(\mathcal{H}, \mathcal{M}, c)$ holds, or there is exactly one mutation in $\mathcal{M}$. Therefore, the core constraints of the ILP include all the constraints in Equations (4), and (10). The number of these core constraints is clearly bounded by $O(mn)$.

3.2 Consistency Constraints

Now we need some constraints to make sure that the assignment of the equation variables are consistent with each other. Consider, for example, three sets of h-variables $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$ that appear in the linear system such that $\mathcal{H}_1 \triangle \mathcal{H}_2 \triangle \mathcal{H}_3 = \emptyset$. (Here, $\triangle$ is the symmetric difference operator.) If $E_{\mathcal{H}_1} = 0$ and $E_{\mathcal{H}_2} = 0$, which are equivalent to $\sum_{h_{i,j} \in \mathcal{H}_1} h_{i,j} = 0$ and $\sum_{h_{i,j} \in \mathcal{H}_2} h_{i,j} = 0$, then we must have $\sum_{h_{i,j} \in \mathcal{H}_3} h_{i,j} = \sum_{h_{i,j} \in \mathcal{H}_1} h_{i,j} + \sum_{h_{i,j} \in \mathcal{H}_2} h_{i,j} = 0$, or equivalently $E_{\mathcal{H}_3} = 0$. The sum of $E_{\mathcal{H}_1}, E_{\mathcal{H}_2}, E_{\mathcal{H}_3}$ must be even. To guarantee such a relation among the three equation variables, we need include the following *consistency constraints*:

$$C(\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3) : \begin{cases} E_{\mathcal{H}_1} + E_{\mathcal{H}_2} + (1 - E_{\mathcal{H}_3}) \geq 1 \\ E_{\mathcal{H}_1} + (1 - E_{\mathcal{H}_2}) + E_{\mathcal{H}_3} \geq 1 \\ (1 - E_{\mathcal{H}_1}) + E_{\mathcal{H}_2} + E_{\mathcal{H}_3} \geq 1 \\ (1 - E_{\mathcal{H}_1}) + (1 - E_{\mathcal{H}_2}) + (1 - E_{\mathcal{H}_3}) \geq 1 \end{cases} \tag{11}$$

These constraints ensure that $(E_{\mathcal{H}_1}, E_{\mathcal{H}_2}, E_{\mathcal{H}_3}) \neq (0,0,1), (0,1,0), (1,0,0), (1,1,1)$, respectively. Therefore, illogical combinations of $E_{\mathcal{H}_1}, E_{\mathcal{H}_2}, E_{\mathcal{H}_3}$ are prohibited, and only legitimate combinations are allowed in a feasible solution.

In general, suppose that $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_r$ is any collection of sets of h-variables that appear in the linear system such that $\mathcal{H}_1 \triangle \mathcal{H}_2 \triangle \cdots \triangle \mathcal{H}_r = \emptyset$. To construct the consistency constraints for their corresponding equation variables, we introduce new variables $S_i = \triangle_{i=1}^{j} \mathcal{H}_i$ and their corresponding variables E_{S_i}. We then construct a series of consistency constraints:

$$C(\mathcal{H}_1, \ldots, \mathcal{H}_r) = C(\mathcal{H}_1, \mathcal{H}_2, S_2) \cup C(S_{r-2}, \mathcal{H}_{r-1}, \mathcal{H}_r) \cup \bigcup_{i=3}^{r-2} C(S_{i-1}, \mathcal{H}_i, S_i) \tag{12}$$

The core constraints and consistency constraints form the complete ILP instance. Note that the number of consistency constraints is generally exponential in n. The following lemma states that these constraints are sufficient for MMHI. Its proof is omitted in this extended abstract.

Lemma 1. *Consider a feasible solution to the (complete) ILP defined above. We can convert the conditional linear system in Section 2.3 to an unconditional linear system using the values of the equation variables in the solution. The linear system must be consistent.*

3.3 The Incremental Approach

Since the complete ILP instance cannot be efficiently constructed in general, we start from an incomplete ILP instance with only the core constraints (no consistency constraints). A standard ILP solver GLPK is invoked to find a solution to the equation variables $E_{\mathcal{H}}$. The equation variable values specifies a set of (unconditional) linear equations from the conditional linear equations in $\mathcal{E}$. We can solve the this system of linear equations by applying Gaussian elimination. However, the linear system may be inconsistent, *i.e.*, there may be a set of equation variables $E_{\mathcal{H}_1}, E_{\mathcal{H}_2}, \ldots, E_{\mathcal{H}_r}$ such that $\triangle_{i=1}^{r}\mathcal{H}_i = \emptyset$ but $\sum_{i=1}^{r} E_{\mathcal{H}_i}$ is odd determined by GLPK. When such an inconsistency occurs, there must be a subset of equations $\left\{\sum_{h\in\mathcal{H}_i} h = c_i\right\}_{i=1}^{r}$ such that $\sum_{i=1}^{r} c_i = 1$ but $\sum_{i=1}^{r}\sum_{h\in\mathcal{H}_r} h = 0$. Hence $\triangle_{i=1}^{r}\mathcal{H}_i = \emptyset$, and we add the consistency constraints shown in Equation (12) to the ILP instance. We then invoke GLPK again. This process is iterated until a solution is found to yield a consistent system of linear equations.

Although in theory this process may take many iterations, more than 95% of the time in our experiment a consistent solution was found in the very first iteration using only the core constraints. Moreover, the process never took more than three iterations in our experiment. This observation can be explained as follows. For each equation constraint in Equation (10), the ILP solver GLPK tends to assign c to the variable $E_{\mathcal{H}}$ given $(\mathcal{M}, \mathcal{H}, c)$ to minimize the number of mutations, if this assignment does not result in conflicting equations. Since the number of mutations is small, most equations should indeed hold. In addition, we usually have a lot of pre-determined μ-variables, which could force GLPK to assign the other variables correctly.

3.4 Phasing, Missing Allele Imputation, and Mutation Detection

Once a consistent (unconditional) linear system is found, solving the system by Gaussian elimination assigns the values of all h-variables. GLPK also assigns the values of all μ-variables in the last iteration. Therefore, we can resolve the p-variables by using the *propagation algorithm* in [12,21]. The basic idea is to propagate known (*i.e.* pre/semi/doubly-determined) p-variable values to undetermined p-variables along the edges in the pedigree by repeatedly applying Equation (3). The p-variables that are left unresolved by the propagation algorithm will be deemed as free in the solution. Note that, the resolved p-variables could allow us to impute missing alleles at some loci (by possibly using some ancestral p-variable and relevant h-variables if necessary), although perhaps not at all loci.

If there are no missing alleles, then the above would produce a consistent solution to the MMHC instance. However, the presence of missing alleles may cause conflict between the assigned values of the μ-variables and those of the

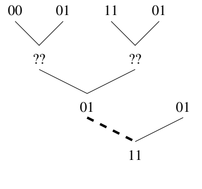

Fig. 3. Missing alleles may prevent us from obtaining path/cycle constraints. In the figure, if there were no missing data, there should have been two path constraints through the dotted line. The μ-variable on the dotted line is free because the two path constraints are not included in the ILP instance.

p-variables and h-variables. This is because some μ-variables do not appear in any conditional equation. These μ-variables only appear in the objective function and the constraints in Equation (4). Let us call this type of μ-variables *free.* (See Figure 3 for an example free μ-variable.) Clearly, the free μ-variables were set to 0 by GLPK to minimize the objective function. This assignment could be in conflict with the p-variable and h-variable values, because their associated path/cycle constraints were not included in the ILP instance. We will try to fix the problem by re-evaluating the free μ-variables using the determined p-variables and h-variables and Equation (3). For any free μ-variable in conflict, we change its value to 1 (which incurs a new mutation).

However, some of these changes might be incorrect (or redundant), and such incorrect changes may potentially lead to other conflicts with the p-variable and h-variable values. When a change leads to more conflicts, we know for sure that the change is wrong (because there can be at most one mutation at the same locus), as stated in the following lemma whose proof is omitted in this extended abstract.

Lemma 2. *If assigning $\mu_{i_1,j_1}[k]$=1 leads to another conflict that forces $\mu_{i_2,j_2}[k]$=1, then both $\mu_{i_1,j_1}[k]$ and $\mu_{i_2,j_2}[k]$ should equal* 0.

Whenever we find two mutations at the same locus, we force their corresponding μ-variables to 0 in the ILP instance (by adding two new constraints), and run GLPK and the propagation algorithm again. Note that, these two μ-variables are no longer viewed as free since they now appear in some constraints in the ILP instance. This process is repeated until all μ-variable values are consistent with the p-variable and h-variable values.

4 Experimental Results

We have implemented our algorithm in C, denoted as MMPhase. A detailed pseudocode of MMPhase is omitted in this extended abstract and will given in the full paper. In this section, we test MMPhase on pedigree data with randomly simulated genotypes, mutations and missing alleles to perform an empirical evaluation of its performance and efficiency. We also compare the speed of MMPhase with that of the straightforward method for solving the MMHC ILP (*i.e.*, running GLPK on all the constraints in a single iteration).

We first compare the speeds of MMPhase and the straightforward method. Since the number of consistency constraints is exponential in the pedigree size n and locus number m in general (even for trees), we implement the straightforward method only for binary trees. When the pedigree is a binary tree, we do not have cycle constraints. For each path constraint along the path between j_1 and j_2, let $\mathcal{H}_1$ be the set of the h-variables on the path from the root of the binary tree to j_1, $\mathcal{H}_2$ the set of the h-variables on the path from the root to j_2, and $\mathcal{H}_3$ the set of h-variables on the path from j_1 to j_2. We put the consistency constraint $C(\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3)$ into the ILP instance. This will provide a sufficient set of consistency constraints which will guarantee a feasible solution to MMHC.

Table 1. The average running times on 100 randomly generated replicates for each pedigree size. The pedigrees are full binary trees.

Pedigree size	Straightforward	Incremental
63	.443s	.144s
127	2.98s	.750s
255	20.3s	4.39s
511	180s	29.0s
1023	29.2m	2.13m

Table 2. The performance of MMPhase under various configurations of the parameters. The default setting includes the pedigree of size 52, 50 marker loci, 10% missing alleles, and 3% mutations. 100 replicated are generated for each configuration of the parameters. Starting from the default setting, we vary the missing rate in (a), the mutation rate in (b), the pedigree in (c), and the number of loci in (d).

Missing rate	Correctly imputed missing alleles	Correctly detected mutations	Correctly phased markers	Running time
0%	—	78.31%	99.98%	2.02s
5%	74.70%	70.20%	98.49%	1.49s
10%	68.97%	62.58%	92.72%	1.24s
20%	69.03%	59.69%	92.75%	.900s

(a)

Mutation rate	Correctly imputed missing alleles	Correctly detected mutations	Correctly phased markers	Running time
1%	73.15%	73.33%	96.75%	1.23s
3%	68.97%	62.58%	92.72%	1.24s
10%	73.11%	69.57%	96.73%	1.47s

(b)

Pedigree size	Correctly imputed missing alleles	Correctly detected mutations	Correctly phased markers	Running time
29	75.34%	52.26%	94.51%	.298s
52	68.97%	62.58%	92.72%	1.24s
128	73.49%	52.11%	93.99%	27.0s

(c)

Locus number	Correctly imputed missing alleles	Correctly detected mutations	Correctly phased markers	Running time
20	73.13%	67.00%	96.78%	.250s
50	68.97%	62.58%	92.72%	1.24s
200	73.09%	65.42%	96.82%	10.8s

(d)

Note that, the number of such consistency constraints is $O(mn)$. Interesting, the incremental approach implemented in MMPhase may theoretically use more consistency constraints in the worst case because of creating redundant variables, although it usually uses a smaller number of consistency constraints in practice. We consider full binary trees of sizes from 63 to 1023, and run both algorithms on 100 randomly generated genotype data with 50 loci, 10% missing alleles,

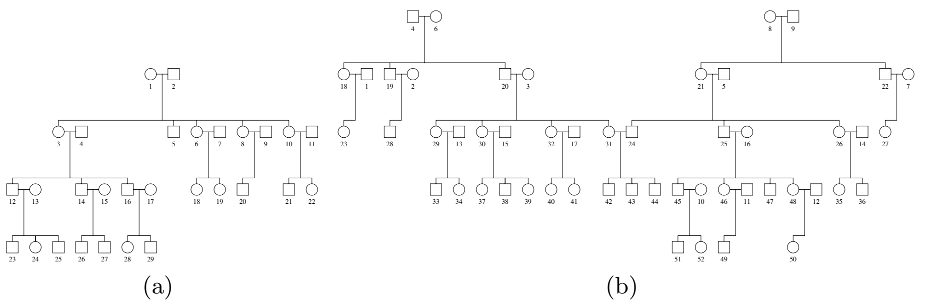

Fig. 4. Three pedigrees are used to test the performance of MMPhase. The first has 29 individuals and is shown in (a). The second has 52 individuals and is shown in (b). The third has 128 individuals and is too large to fit in the page.

and 3% mutations (*i.e.* 3% of the loci are mutated in inheritance). (Actually, haplotypes are generated in the simulations and then converted to genotypes as an input of the algorithms.) Table 1 shows the average running times of both algorithms on each full binary tree. We observe that MMPhase is much faster than the straightforward method, and the speedup ratio increases as the pedigree size gets bigger. For example, the ratio is about 14 on full binary trees of size 1023. We also observe that the solutions from both algorithms are sometimes slightly different but they always require the same number of mutations which is smaller than the actual number of mutations simulated (the detailed results are not shown).

Next we test the performance of MMPhase in terms of the percentage of correctly phased markers, the percentage of correctly imputed missing alleles, and the percentage of correctly detected mutations. (A simulated mutation is correctly detected if there is an inferred mutation that coincides with its location exactly.) We use three real human pedigrees from the literature as shown in Figure 4. 100 replicates of genotype data is simulated on each of these pedigrees with each of several configurations of the number of marker loci, the missing allele rate and the mutation rate. Our default setting of simulation uses the pedigree of size 52, 50 marker loci, 10% missing alleles, and 3% mutations. To observe how each of these parameters affects the performance MMPhase, we vary one parameter at a time in the test. Table 2 illustrates the test results. We observe that, as shown in Table 2(a), higher missing rates lead to faster performance since fewer path/cycle constraints are added to the ILP instance. Not surprisingly, higher missing rates also result in fewer correctly detected mutations and fewer correctly phased markers. Table 2(b) shows that the performance is not very sensitive to the mutation rate. Table 2(c) and Table 2(d) show that the pedigree and number of marker loci mainly affect the running time.

In conclusion, MMPhase is very efficient and can infer haplotypes very accurately. It can also recover most of the mutations and missing alleles correctly.

Note that, our criterion for correctly detecting a mutation is very stringent since in some cases the mutation could be shifted in the pedigree without affecting the feasibility of the solution (especially when missing alleles are present).

5 Concluding Remarks and Acknowledgements

Genotyping errors are very common in practice and can easily be confused with mutations. It would be interesting to extend the method to deal with both mutations and genotyping errors. The research is supported in part by NIH grant LM008991 and NSF grant IIS-0711129.

References

1. Abecasis, G.R., Cherny, S.S., Cookson, W.O., Cardon, L.R.: Merlin — rapid analysis of dense genetic maps using sparse gene flow trees. Nature Genetics 30(1), 97–101 (2002)
2. Albers, C.A., Heskes, T., Kappen, H.J.: Haplotype inference in general pedigrees using the cluster variation method. Genetics 177(2), 1101–1116 (2007)
3. Badaeva, T.N., Malysheva, D.N., Korchagin, V.I., Ryskov, A.P.: Genetic variation and *De Novo* mutations in the parthenogenetic caucasian rock lizard *Darevskia unisexualis*. PLoS ONE 3(7), e2730 (2008)
4. Baruch, E., Weller, J.I., Cohen-Zinder, M., Ron, M., Seroussi, E.: Efficient inference of haplotypes from genotypes on a large animal pedigree. Genetics 172(3), 1757–1765 (2006)
5. Ellegren, H.: Microsatellite mutations in the germline: Implications for evolutionary inference. Trends in Genetics 16(12), 551–558 (2000)
6. Gusfield, D.: Inference of haplotypes from samples of diploid populations: Complexity and algorithms. J. Computational Biology 8(3), 305–323 (2001)
7. Kimura, M., Crow, J.F.: The number of alleles that can be maintained in a finite population. Genetics 49, 725–738 (1964)
8. Lander, E.S., Green, P.: Construction of multilocus genetic linkage maps in humans. In: Proc. of the National Academy of Sciences. Genetics, vol. 84, pp. 2363–2367 (1987)
9. Li, J., Jiang, T.: Efficient inference of haplotypes from genotypes on a pedigree. J. Computational Biology 1(1), 41–69 (2003)
10. Li, J., Jiang, T.: Computing the minimum recombinant haplotype configuration from incomplete genotype data on a pedigree by integer linear programming. J. Computational Biology 12(6), 719–739 (2005)
11. Li, J., Jiang, T.: A survey on haplotype algorithms for tightly linked markers. J. Bioinformatics and Computational Biology 6(1), 241–259 (2008)
12. Liu, L., Jiang, T.: Linear-time reconstruction of zero-recombinant mendelian inheritance on pedigrees without mating loops. Genome Informatics 19, 95–106 (2007)
13. Niu, T., Qin, Z.S., Xu, X., Liu, J.S.: Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. Am. J. Hum. Genet. 70(1), 157–169 (2002)
14. Olson, T.M., Doan, T.P., Kishimoto, N.Y., Whitby, F.G., Ackerman, M.J., Fananapazir, L.: Inherited and *de novo* mutations in the cardiac actin gene cause hypertrophic cardiomyopathy. J. Molecular and Cellular Cardiology 32(9), 1687–1694 (2000)

15. Qian, D., Beckmann, L.: Minimum-recombinant haplotyping in pedigrees. Am. J. Hum. Genet. 70(6), 1434–1445 (2002)
16. Sobel, E., Lange, K., O'Connell, J.R., Weeks, D.E.: Haplotyping algorithms. In: Speed, T., Waterman, M.S. (eds.) Genetic Mapping and DNA Sequencing. IMA Volumes in Mathematics and its Applications, vol. 81, pp. 89–110. Springer, Heidelberg (1996)
17. Stephens, M., Smith, N.J., Donnelly, P.: A new statistical method for haplotype reconstruction from population data. Am. J. Hum. Genet. 68(4), 978–989 (2001)
18. Tapadar, P., Ghosh, S., Majumder, P.P.: Haplotyping in pedigrees via a genetic algorithm. Human Heredity 50(1), 43–56 (2000)
19. The Internaltional HapMap Consortium. The international HapMap project. Nature 426, 789–796 (2003)
20. Wang, S., Kidd, K.K., Zhao, H.: On the use of DNA pooling to estimate haplotype frequencies. Genetic Epidemiology 24(1), 74–82 (2003)
21. Xiao, J., Liu, L., Xia, L., Jiang, T.: Fast elimination of redundant linear equations and reconstruction of recombination-free mendelian inheritance on a pedigree. In: 18th Annual ACM-SIAM Symposium on Descrete Algorithms, pp. 655–664 (2007)
22. Yang, Y., Zhang, J., Hoh, J., Matsuda, F., Xu, P., Lathrop, M., Ott, J.: Efficiency of single-nucleotide polymorphism haplotype estimation from pooled DNA. Proc. of the National Academy of Sciences 100, 7225–7230 (2002)
23. Zhang, K., Sun, F., Zhao, H.: HAPLORE: A program for haplotype reconstruction in general pedigrees without recombination. Bioinformatics 21(1), 90–103 (2005)