Linear-Time Haplotype Inference on Pedigrees without Recombinations

M. Y. Chan[1], Wun-Tat Chan[1], Francis Y. L. Chin[1]*, Stanley P. Y. Fung[2], and Ming-Yang Kao[3]

[1] Department of Computer Science, University of Hong Kong, Hong Kong
{mychan, wtchan, chin}@cs.hku.hk
[2] Department of Computer Science, University of Leicester, Leicester, UK
pyfung@mcs.le.ac.uk
[3] Department of Electrical Engineering and Computer Science, Northwestern University, USA
kao@cs.northwestern.edu

Abstract. In this paper, a linear-time algorithm, which is optimal, is presented to solve the haplotype inference problem for pedigree data when there are no recombinations and the pedigree has no mating loops. The approach is based on the use of graphs to capture SNP, Mendelian and parity constraints of the given pedigree.

1 Introduction

The modeling of human genetic variation is critical to the understanding of the genetic basis for complex diseases. *Single nucleotide polymorphisms* (SNPs)[6] are the most frequent form of this variation, and it is useful to analyze *haplotypes*, which are sequences of linked SNP genetic markers (small segments of DNA) on a single chromosome. In diploid organisms, such as humans, chromosomes come in pairs, and experiments often yield *genotypes*, which blend haplotypes for the chromosome pair. This gives rise to the problem of inferring haplotypes from genotypes.

Before defining our problem, some preliminary definitions are needed. The physical position of a marker on a chromosome is called a *locus* and its state is called an *allele*. Without loss of generality, the alleles of a *biallelic* SNP can be denoted by 0 and 1, and a haplotype with m loci is represented as a length-m string in $\{0,1\}^m$, and a genotype as a length-m string in $\{0,1,2\}^m$. Haplotype pair $\langle h_1,\, h_2\rangle$ is *SNP-consistent* with genotype g if where the two alleles of h_1 and h_2 are the same at the same locus, say 0 (respectively 1), the corresponding locus of g is also 0 (1), which denotes a *homozygous* locus; otherwise, where the two alleles of h_1 and h_2 are different, the corresponding locus of g is 2, which denotes a *heterozygous* locus (i.e. SNP). A genotype with s heterozygous loci can have 2^{s-1} SNP-consistent haplotype solutions. For example, genotype

* This research was supported by Hong Kong RGC Grant HKU-7119/05E and HKU Strategic Research Team Fund

$g = 012212$ with $s = 3$ has four SNP-consistent haplotype pairs: $\{\langle 01\underline{11}1\underline{1}, 01\underline{00}1\underline{0}\rangle, \langle 01\underline{11}1\underline{0}, 01\underline{00}1\underline{1}\rangle, \langle 01\underline{10}1\underline{1}, 01\underline{01}1\underline{0}\rangle, \langle 01\underline{10}1\underline{0}, 01\underline{01}1\underline{1}\rangle\}$.

A *pedigree* is a fundamental connected structure used in genetics. Figure 1 shows the pictorial representation of a pedigree with 4 nodes, with a square representing a male node and a circle representing a female node and children placed under their parents: in particular, a *father* (node F), a *mother* (node M) and two *children* (son node S and daughter node D). F-M-S (also F-M-D) is a *father-mother-child trio* or simply *trio*. Furthermore, each individual node in the pedigree is associated with a genotype. We assume that there are no *mating loops*, i.e., no marriages between descendants of a common ancestor, in the pedigree.

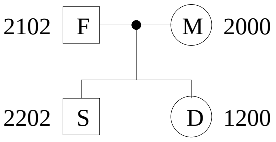

Fig. 1. Example of a pedigree with 4 nodes.

A *Consistent Haplotype Configuration* (with no recombinations) for a given pedigree is an assignment of a pair of haplotypes to each individual node such that (i) all the haplotype pairs are SNP-consistent with their corresponding genotypes and (ii) the haplotypes of each child are *Mendelian-consistent*, i.e. one of the child's haplotype is exactly the same as one of its father's and the other is the same as one of its mother's.

Haplotyping Pedigree Data (with No Recombinations) Problem (HPD-NR): Given a pedigree P where each individual node of P is associated with a genotype, find a consistent haplotype configuration (CHC) for P. □

Wijsman [8] proposed a 20-rule algorithm, and O'Connell [5] described a genotype elimination algorithm, both of which can be used for solving the HPD-NR problem. Li and Jiang [2] formulated the problem as an $mn \times mn$ matrix and solved HPD-NR by Gaussian elimination which could be solved in polynomial time ($O(m^3n^3)$), where n is the number of individuals in the pedigree and m is the number of loci for each individual. Xiao, Liu, Xia and Jiang [9] later improved this to $O(mn^2 + n^3 \log^2 n \log\log n)$. For the case without mating loops, their algorithm runs in $O(mn^2 + n^3)$ time. In this paper, we propose a new 4-stage algorithm that can either find a CHC solution or report "no solution" in optimal $O(mn)$ time when there are no mating loops. Due to space constraints, some proofs are omitted from this version.

2 The Algorithm

2.1 Stage 1

Definition 1. *If there exists a father F, mother M and two children C_1 and C_2 in the pedigree and two locus i and j such that i and j are heterozygous loci for F, M and C_1 but are homozygous and heterozygous, respective, for C_2, then we say that the pedigree has a* **family problem**. □

Stage 1A - Checking for family problems: Since a pedigree with a family problem has no CHC solution, our algorithm begins by checking for family problems. Only if there are no family problems will the algorithm continue; otherwise, "no solution" is reported. □

Stage 1B – Generation of vector-pairs: For each trio in the given pedigree, let the respective genotypes of the father F, the mother M and the child C be: $x_1x_2 \ldots x_m$ and $y_1y_2 \ldots y_m$ and $z_1z_2 \ldots z_m$ where x_i, y_i, $z_i \in \{0, 1, 2\}$. We determine a pair of vectors (or vector-pair) each for the father, the mother and the child, namely: $\langle f_1, f_2 \rangle$, $\langle m_1, m_2 \rangle$ and $\langle c_1, c_2 \rangle$, respectively, where $f_1 = x_{1,1}x_{1,2} \ldots x_{1,m}$ and $f_2 = x_{2,1}x_{2,2} \ldots x_{2,m}$; $m_1 = y_{1,1}y_{1,2} \ldots y_{1,m}$ and $m_2 = y_{2,1}y_{2,2} \ldots y_{2,m}$; $c_1 = z_{1,1}z_{1,2} \ldots z_{1,m}$ and $c_2 = z_{2,1}z_{2,2} \ldots z_{2,m}$. The vector-pairs are determined in the following manner.

1. For each locus i, for f_1 and f_2:
   (a) If $x_i = 0$ then $x_{1,i} = x_{2,i} = 0$.
   (b) If $x_i = 1$ then $x_{1,i} = x_{2,i} = 1$.
   (c) If $x_i = 2$ and $z_i = 0$ then $x_{1,i} = 0$ and $x_{2,i} = 1$.
   (d) If $x_i = 2$ and $z_i = 1$ then $x_{1,i} = 1$ and $x_{2,i} = 0$.
   (e) If $x_i = 2$ and $z_i = 2$ and $y_i = 0$ then $x_{1,i} = 1$ and $x_{2,i} = 0$.
   (f) If $x_i = 2$ and $z_i = 2$ and $y_i = 1$ then $x_{1,i} = 0$ and $x_{2,i} = 1$.
   (g) If $x_i = 2$ and $z_i = 2$ and $y_i = 2$ then $x_{1,i} = ?$ and $x_{2,i} = ?$.
2. m_1 and m_2 are similarly determined.
3. We assume C inherits f_1 from F and m_1 from M and thus $\langle c_1, c_2 \rangle = \langle f_1, m_1 \rangle$. Check if $\langle c_1, c_2 \rangle$ is consistent with C's genotype $z_1z_2 \ldots z_m$. If not, report "no solution". □

Observe that if a particular node N in the pedigree belongs to k different trios, then k vector-pairs, or $2k$ vectors, will be created for N in Stage 1. Let Φ(N) be the multiset comprised of these k vector-pairs. It is sometimes convenient to refer to the vectors rather than the vector-pairs. Thus, we let Γ(N) be the multiset of $2k$ vectors, containing the two vectors of each vector-pair in Φ(N). Note that we can define SNP-consistency and Mendelian-consistency in terms of vector-pairs.

SNP-Consistency Condition: *SNP-consistency is said to be maintained* iff, for all nodes N in the pedigree, each vector-pair in Φ(N) is SNP-consistent with N's genotype. Vector-pair $\langle h_1, h_2 \rangle$ is said to be *SNP-consistent* with genotype g if h_1 and h_2 are both 0 (respectively 1) at the same locus, the corresponding locus of g is also 0 (1); otherwise, if h_1 is 0 (respectively 1) and h_2 is 1 (0) at the same locus, the corresponding locus of g is 2 (2). □

Mendelian-Consistency Condition [1, 7]: *Mendelian-consistency is said to be maintained* iff, for all nodes N in the pedigree, if N is a child in a trio comprised of F, M and N, then Φ(N) contains a vector-pair $\langle c_1, c_2 \rangle = \langle f_1, m_1 \rangle$ where $f_1 \in \Gamma$(F) and $m_1 \in \Gamma$(M). □

Stage 1C - Initial construction of $G = (V, E)$: Let V be the multiset of all the vectors created in Stage 1B and E be the set of *red* and *brown* edges defined below.

1. A **red** edge will be introduced to join the two vectors of each vector-pair generated in Stage 1A and indicates that a ? appearing at locus i of both vectors must be resolved differently in the later stages of the algorithm (the two vectors can be different or the same at other locus positions depending on whether the genotype has a 2 or not at that locus). *[SNP-consistency]*
2. For each F-M-C trio, let $\langle f_1, f_2\rangle$, $\langle m_1, m_2\rangle$ and $\langle c_1, c_2\rangle$ be vector-pairs in Φ(F), Φ(M) and Φ(C), respectively, associated with this trio. Two **brown** edges will be introduced, one connecting $c_1$and f_1, and the other connecting $c_2$and m_1. A brown edge between two vectors means that the two vectors must be the same at all locus positions. *[Mendelian-consistency]* □

Example 1: Consider the pedigree with F (father), M (mother), S (son), D (daughter) shown in Figure 1. Stage 1 produces the following graph G of 12 vertices and 10 edges (6 red and 4 brown), comprised of two connected components.

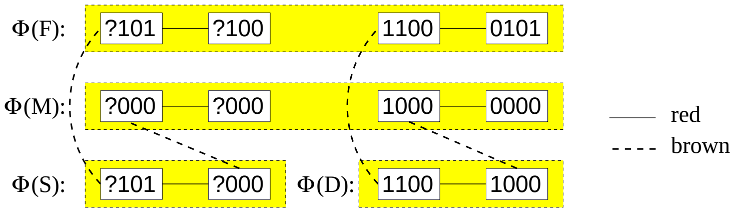

Fig. 2. Graph G for Example 1.

Definition 2. *For any loci in a connected component $\mathcal{G}$ of G, we say*

1. *Locus i is* **resolved** *in $\mathcal{G}$ iff all vectors in $\mathcal{G}$ have 0 or 1 at locus i.*
2. *Locus i is* **unresolved** *in $\mathcal{G}$ iff all vectors in $\mathcal{G}$ have ? at locus i.*
3. *Otherwise, locus i is a mix of ? and non-? at i.* □

In Example 1, the connected component for trio F-M-S has one unresolved locus (locus 1) and three resolved loci (locus 2, 3 and 4). Meanwhile, the component for trio F-M-D has no unresolved loci and four resolved loci (locus 1, 2, 3 and 4).

Lemma 1. *The time complexity of Stage 1 (Stage 1A, 1B and 1C) is $O(mn)$, where n is the number of nodes in the pedigree and m is the number of loci in each genotype. Furthermore, after Stage 1, all loci are either resolved or unresolved in each connected component of G, and G has $O(n)$ nodes and edges.* □

In Stages 2 and 3, no vector-pairs will be added to or deleted from each Φ(N) and the 0's and 1's of Stage 1 will remain as they are (unchanged). The unresolved loci of each component of G will become resolved with SNP-consistency and Mendelian-consistency maintained, and components of G will be repeatedly merged with the addition of connecting **green** (added in Stage 2) or **white** (added in Stage 3) edges until G evolves into being a single connected component. Each green or white edge is added between two vectors belonging in the same $\mathit{\Gamma}$(N). This structured way of adding edges to make G connected can be done given Lemma 2 below.

Lemma 2. *If G has more than one connected component, then there exists a $\mathit{\Phi}$(N) for some N such that there are two vector-pairs in $\mathit{\Phi}$(N) which belong to two different components.*

Proof. Suppose to the contrary that, for all N, the vector-pairs in $\mathit{\Phi}$(N) are all connected. We make use of the fact that the brown edges in G preserve the connectivity of any two nodes in the pedigree, which we have assumed to be connected. Therefore, if vector-pairs in $\mathit{\Phi}$(N) are all connected for all N, then all vectors are connected together in a single connected component, which contradicts the assumption that G has more than one connected component. □

As loci are resolved, each multiset $\mathit{\Phi}$(N) may contain one or more copies of more than one unique vector-pair. However, by the time all loci are resolved, for all nodes N, each multiset $\mathit{\Phi}$(N) must contain k copies of one unique vector-pair $\langle h_1, h_2\rangle$, which represents the haplotype-pair in the CHC for N, where k is the number of trios to which N belongs. We need an additional condition:

Endgame-Consistency Condition: *Endgame-consistency is said to be maintained* iff, for all nodes N is the pedigree, N is Endgame-consistent. Node N is said to be *Endgame-consistent* if there does not exist vector-pairs $\langle u_1, u_2\rangle$, $\langle v_1, v_2\rangle \in \mathit{\Phi}$(N) such that the vector values at some heterozygous locus i and j $(i \neq j)$ for u_1, u_2, v_1 and v_2 are a permutation of the four possibilities: 00, 01, 10 and 11; and *Endgame-inconsistent* otherwise. A connected component $\mathcal{G}$ of graph G is said to be *Endgame-consistent* if there does not exist a node N and vector-pairs $\langle u_1, u_2\rangle$, $\langle v_1, v_2\rangle$ in both $\mathit{\Phi}$(N) and $\mathcal{G}$ such that the vector values at some heterozygous locus i and j $(i \neq j)$ for u_1, u_2, v_1 and v_2 are a permutation of the four possibilities: 00, 01, 10 and 11; and *Endgame-inconsistent* otherwise. □

Our algorithm achieves a solution if, at the end of Stage 4, (a) graph G comprises a single connected component; (b) all loci are resolved in G; and (c) SNP-consistency, Mendelian-consistency and Endgame-consistency are maintained. However, our algorithm might report "no solution" if some N is Endgame-inconsistent before the end of Stage 4.

2.2 Stage 2

We begin by defining an important subroutine called LOCUS_RESOLVE. LOCUS_RESOLVE($\mathcal{G}$, i, u, x) will resolve all ?'s at an unresolved locus i in a connected component $\mathcal{G}$ (of G) starting with resolving the ? at locus i of vector u in $\mathcal{G}$ to $x \in \{0, 1\}$ in a manner consistent with red and non-red edges.

```
LOCUS_RESOLVE($\mathcal{G}, i, u, x$):
 1. Let vector $u = u_1 u_2 \ldots u_m$. Set $u_i \leftarrow x$
```

```
For each edge $e = (u, v)$:
    Let vector $v = v_1 v_2 \ldots v_m$.
    If $v_i = ?$ then
        If $e$ is a red edge then LOCUS_RESOLVE($\mathcal{G}$, $i$, $v$, $1 - x$)
        else LOCUS_RESOLVE($\mathcal{G}$, $i$, $v$, $x$)                □
```

The idea of Stage 2 is to add $O(n)$ green edges to connect components of G together, where green edges are like brown edges requiring that the ?s in the two vectors connected by the edge to be resolved the same. The way in which green edges are added respects Endgame-consistency. In particular, green edges are added to connect two unconnected vectors that have the value 0 at heterozygous locus i.

Stage 2 – Adding Green Edges: For each locus i do the following:

1. For each node N, if locus i is heterozygous in N, (a) let $u = u_1 u_2 \ldots u_m$ in Γ(N) such that $u_i = 0$ (if any); and (b) for each other vector $v = v_1 v_2 \ldots v_m$ in Γ(N) such that $v_i = 0$ do the following:
   (a) For each locus j such that $u_j \in \{0,1\}$ and $v_j = ?$, run LOCUS_RESOLVE (G_v, j, v, u_j). In so doing, we say that we **use $\boldsymbol{u}$ to resolve all unresolved loci of $\boldsymbol{G_v}$**.
   (b) Likewise, for each locus j such that $v_j \in \{0,1\}$ and $u_j = ?$, run LOCUS_RESOLVE(G_u, j, u, v_j). Thus, we **use $\boldsymbol{v}$ to resolve all unresolved loci of $\boldsymbol{G_u}$**.
   (c) Add a green edge joining u and v.
2. Make G acyclic, by removing green edges only. □

Lemma 3. *The time complexity of Stage 2 is $O(mn)$. Furthermore, after Stage 2, all loci are either resolved or unresolved in each connected component of G, and G has $O(n)$ nodes and edges.*

Proof. There are two aspects for the time complexity of Stage 2. Firstly, only unresolved loci in each component are considered, and thus a locus, once resolved, will not be considered again even upon the component's subsequent joining with other components by green edges. In this way, $O(mn)$ time complexity can be achieved. Secondly, when heterozygous locus i is considered, at most $n-1$ green edges will be added to G and thus G will still have $O(n)$ edges. Step 2 is intended to prevent an explosion of green edges by eliminating any cycles among vectors in Γ(N) by removing green edges and can be done in $O(n)$ time by a traversal of G and is only done once for each locus. Note that, after Stage 2, there may still exists unconnected vectors u and v in Γ(N) with $u_i = v_i = 0$ for some heterozygous locus i in N; such u and v will become properly connected in Stage 3. □

Stage 2 ensures that each connected component has only resolved and unresolved loci. This property is important. Lemma 4 essentially tells us that we can arbitrarily resolve unresolved loci in any such component of G, and it will not affect Endgame-consistency in the sense that no matter how the unresolved loci are resolved, either Endgame-consistency will be maintained or not maintained within that component. Stage 1A and Stage 2 combined ensure the mother-father property of Lemma 5.

Lemma 4. *If a component $\mathcal{G}$ (of G) has only resolved and unresolved loci, then all possible ways of resolving ?'s in vectors in $\mathcal{G}$ such that SNP-consistency and Mendelian-consistency are maintained will either all make $\mathcal{G}$ Endgame-consistent or all make $\mathcal{G}$ Endgame-inconsistent.*

Proof. Consider a particular resolution of ?'s in the vectors in $\mathcal{G}$ such that SNP-consistency and Mendelian-consistency are maintained. Suppose Endgame-inconsistency occurs at node N, i.e. there exist two vector-pairs $\langle x_1, x_2\rangle$, $\langle y_1, y_2\rangle \in \Phi(\mathrm{N})$. We can assume, without loss of generality, that the value at some heterozygous locus i and j ($i \neq j$) for x_1, x_2, y_1 and y_2 are 00, 11, 01 and 10 respectively. Consider the following three cases for the state of locus i and j prior to the resolution:

Case 1: *Suppose locus i and j were both unresolved in $\mathcal{G}$.* Then, for all other possible resolutions, the values at locus i and j for x_1, x_2, y_1 and y_2 would either be 00, 11, 01 and 10 respectively, or 11, 00, 10 and 01 respectively, and Endgame-consistency would also be violated.

Case 2: *Suppose only one of locus i and j was unresolved, say i, in $\mathcal{G}$.* Then, for all other possible resolutions, the values at locus i and j for x_1, x_2, y_1 and y_2 would either be 00, 11, 01 and 10 respectively, or 10, 01, 11 and 00 respectively, and Endgame-consistency would also be violated.

Case 3: *Suppose both locus i and j were not unresolved (i.e., resolved).* Then, the Endgame-inconsistency existed prior to any resolution of ?'s. □

Lemma 5. *Suppose (a) M and F are the mother and father of two unconnected trios in G after Stage 2 and (b) the given pedigree has no family problems. Then, for all possible way of resolving ?s in vectors in the two trios such that SNP-consistency and Mendelian-consistency are maintained, M and F are either both Endgame-consistent or both Endgame-inconsistent.*

Proof. Suppose F is Endgame-inconsistent. Without loss of generality, let the values at locus i and j for x_1, x_2, y_1 and y_2 be 00, 11, 01 and 10 respectively where $\langle x_1, x_2\rangle$, $\langle y_1, y_2\rangle \in \ \phi(\mathrm{F})$. This means locus i and j are heterozygous loci for F. Since the two trios are unconnected by a green edge, locus i and j are also heterozygous for M also. Let C_1 and C_2 be the two respective children of F connected to $\langle x_1, x_2\rangle$ and $\langle y_1, y_2\rangle$ by a brown edge. In the absence of family problems and green edges connecting the two trios, there are only three cases to consider: (i) when locus i and j are both heterozygous for both C_1 and C_2; (ii) when locus i and j are both heterozygous for C_1 and both homozygous for C_2; and (iii) when locus i is heterozygous for C_1 and homozygous for C_2 while locus j is homozygous for C_1 and heterozygous for C_2. It can be readily shown that in all three cases, M would also be Endgame-inconsistent. □

2.3 Special Case of a Connected Graph

Let us consider the special case where G becomes a connected graph (i.e. a single connected component) after Stage 2. By Lemma 3, we are left only with at most two kinds of locus in G: resolved and unresolved. To resolve all unresolved loci in G (if any), we do the following. Arbitrarily pick a vector u of G. For all unresolved locus i, we simply run LOCUS_RESOLVE(G, i, u, 0). Note that running LOCUS_RESOLVE(G, i, u, 1) would have worked equally well (Lemma 4), the effect being all 1's become 0 and all 0's become 1 at locus i and gives another solution. Finally, we check that that all N are Endgame-consistent and report "no solution" if any N were Endgame-inconsistent. This procedure for dealing with G when G is a single connected component will later be called Stage 4.

Lemma 6. *If G is a connected graph after Stage 2, we can either achieve a solution that represents a CHC for the given pedigree, or report "no solution" when there is no CHC for the pedigree, in $O(mn)$ time.*

Proof. By Lemma 4, we do not have to try all possible resolutions; one will do. The time complexity of resolving the remaining k unresolved loci in the manner described above is $O(kn)$ since LOCUS_RESOLVE runs in $O(n)$ time. Checking all N for Endgame-consistency can be done in $O(mn)$ time. □

Lemma 7. *Suppose G is a connected graph after Stage 2. If there exists a CHC solution, there are 2^s different CHC solutions, where s is the number of unresolved loci in G, unless every node in the pedigree has exactly s heterozygous loci in which case there are 2^{s-1} different CHC solutions.*

Proof. If there is a CHC solution, it is easy to see that it will remain a solution if all values at a particular unresolved locus were reversed (i.e. 0 changed to 1 and vice versa) because SNP-consistency, Mendelian-consistency and Endgame-consistency will be maintained. Thus, there are 2^s possible CHC solutions altogether, as long as there exists at least one node with more than s heterozygous loci. However, when each node in the pedigree has exactly s heterozygous loci, i.e. all the other loci are homozygous, the number of different CHC solutions is 2^{s-1}. □

2.4 Stage 3

After Stage 2, suppose G is left with r connected components where $r > 1$, with each component having only resolved and unresolved loci. The idea of Stage 3 is to connect components of G together so that a single connected component results. After G becomes a single connected component, we can continue in the manner described in the previous section for a single connected component. Note that white edges will be treated as "non-red" edges by LOCUS_RESOLVE.

As it turns out, we can connect components in a structured way with the help of a ***support graph H***. This we do in Stage 3A.

Stage 3A – Constructing Support Graph *H*:

1. For each node N in the pedigree, if N is unmarried, Γ(N) cannot intersect with more than one connected component of G. Nothing is added to H. Otherwise, suppose N is married to M in the pedigree. Let G_N denote the set of connected components in G that intersect Γ(N) but not Γ(M). Similarly, G_M denote those that intersect Γ(M) but not Γ(N), and G_{MN} denote those that intersect both Γ(M) and Γ(N). Now,
   (a) Pick a vector from Γ(N) from each connected component in $G_N \cup G_{MN}$. Connect the k chosen vectors with $k-1$ edges.
   (b) Next, pick a vector from Γ(M) from each connected component in G_M. Connect them to one of the vectors in Γ(M) from a connected component in G_{MN}.
2. Next, we introduce $k'-1$ edges to connect up the k' vectors in H that are in the same component of G, and for each such edge (u, v) introduced, we label the edge with 0 if there is a path with an even number of red edges between u and v in G; otherwise, we label it with 1.

Lemma 8. *If there are no mating loops in the pedigree, H is acyclic.*

Proof. We claim that, if there are no mating loops (cycles) in the pedigree, any two components both intersect the Γ of at most two nodes. Furthermore, if there are two such nodes, they are the parents within two unconnected trios. This being the case, by making sure there are no cycles between a node and its spouse in H, as we have done in Step 1, there are no cycles in H. To prove the claim, we make use of the fact that the brown edges in G preserve and reflect the connectivity of any two nodes in the pedigree. □

Lemma 9. *H has $O(n)$ edges, and can be constructed in $O(n)$ time.* □

The idea is that we will label each edge of support tree H with 0 and 1. Some edges have been labeled in Stage 3A and others have not. We are mainly interested in the label of edge (u, v) in H where u and v are unconnected in G. Such a labeling will be done in Stage 3C. If the label is 0, then we would connect (unconnected) u and v with a white edge in G. Otherwise, we would instead connect u and the vector that is connected to v by a red edge. This is how H is used. Note that, a CHC solution of the pedigree corresponds a labeling of the edges of H. Our challenge is to finding that labeling.

In order to assist the labeling, we construct a ***parity constraint graph J***, which is constructed in Stage 3B. One of the essential differences between H and J is that H shows connections between "neighboring" components while J captures all parity constraints between far-apart components.

Stage 3B – Construct parity constraint graph J:

1. Nodes in J are the same as the nodes in H.
2. Add an edge between two vectors u and v in J if (u, v) is labeled in H. Furthermore, the label of this edge in J is the same as its label in H.
3. If there is a path between two vectors u and v in H and a heterozygous locus i such that u and v are resolved (has 0 or 1) at locus i but all other vectors

(if any) in the path are unresolved at locus i, add an edge (u, v) labeled L between u and v in J, where L is 1 if u and v are resolved differently at locus i and 0 otherwise, provided there is no such edge already in J. Note that there may still be two edges between any two pairs of vectors u and v in J, one labeled 0 and the other labeled 1, which is an odd cycle.

4. Check that all cycles in J have an even number of edges labeled 1. Report "no solution" and stop if there is a cycle in J with an odd number of edges labeled 1.
5. Let graph K be a copy of graph J. Note that K is not necessarily connected. To make K connected, we add edge (u, v) to K when u and v are in different components in K where (u, v) is an edge in H. This is always possible because H is a connected graph and K and H have the same set of vectors as nodes. We arbitrarily label this edge with 0 and call the corresponding edge in H a ***free edge*** because we have the freedom to label (u, v) with 1 instead. We continue adding edges until K is connected. □

Lemma 10. *If H has no cycles but J has an odd cycle, then there is no CHC solution.* □

Lemma 11. *K has at most $O(mn)$ edges and can be constructed in $O(mn)$ time.* □

Stage 3C – Complete labeling of H:

1. Traverse K, computing, for each node v in K, whether the number of 1-labeled edges in the path from a fixed node t in K is odd or even, i.e. parity.
2. For each unlabeled edge (u, v) in H: if u and v have same parity in K then label edge (u, v) in H with 0; else with 1. □

Lemma 12. *All edges in H can be labeled with 0 or 1 in $O(mn)$ time in Stage 3C, and the labels in H are consistent with the parity constraints specified in J in the sense that the parity between any two vectors u and v specified in J is consistent with the number of 1-label edges in the path between u and v in H.* □

Lemma 13. *Suppose the pedigree has a CHC solution, which corresponds to a labeling of edges in H where free edge e is labeled $\alpha \in \{0, 1\}$. Then, changing the label on e to $1 - \alpha$ will result in a labeling that also corresponds to a CHC solution.* □

Stage 3D – Adding White Edges to G: For each edge (u, v) in H where u is in say component G_u and v in G_v:

1. If edge is labeled 1 then let $x \leftarrow$ vector adjacent to v by red edge else $x \leftarrow v$.
2. Add white edge between u and x.
3. Use u to resolve unresolved loci in G_v.
4. Use x to resolve unresolved loci in G_u.
5. G now has one less component.

Lemma 14. *Stage 3D can be done in $O(mn)$ time, and after Stage 3D, G will be a single connected component with only unresolved and resolved loci.* □

Lemma 15. *If the pedigree has a CHC solution, Stage 3D maintains Endgame-consistency.*

Proof. Suppose, to the contrary, that some node N becomes Endgame-inconsistent after Stage 3D. Without loss of generality, let the values at locus i and j for x_1, x_2, y_1 and y_2 be 00, 11, 01 and 10, respectively, where $\langle x_1, x_2\rangle$, $\langle y_1, y_2\rangle \in \Phi$(N). We say that the two vectors are Endgame-inconsistent.

Consider the situation prior to Stage 3D. Since the pedigree has a CHC solution, given Lemma 4, $\langle x_1, x_2\rangle$ and $\langle y_1, y_2\rangle$ must belong to the different components. Now suppose $\langle x_1, x_2\rangle$ and $\langle y_1, y_2\rangle$ become connected during Stage 3D, in particular, after the addition of a white edge e. Before the addition of white edge e, suppose $\langle x_1, x_2\rangle$ belonged to component G_1 and $\langle y_1, y_2\rangle$ belonged to component G_2. There are four cases to consider:

Case 1: *e connects $\langle x_1, x_2\rangle$ and $\langle y_1, y_2\rangle$.* White edge e corresponds to an edge in H that is labeled with a unique parity. Suppose e connects x_1 and y_1 and is labeled 0. This white edge will make x_1 and y_1 equal and therefore the value of locus i and j cannot possibly become 00 for x_1 and 01 for y_1.

Case 2: *e connects $\langle x_3, x_4\rangle$ in G_1 and $\langle y_3, y_4\rangle$ in G_2 where $\langle x_3, x_4\rangle$ and $\langle y_3, y_4\rangle$ $\in \Phi$(N).* Since pedigree has a CHC solution and G_1 has only resolved and unresolved loci, according to Lemma 4, G_1 must be Endgame-consistent. This implies that $\langle x_1, x_2\rangle$ and $\langle x_3, x_4\rangle$, which are in G_1, are Endgame-consistent. Likewise, $\langle y_1, y_2\rangle$ and $\langle y_3, y_4\rangle$ must also be Endgame-consistent. Because of the argument in Case 1, $\langle x_3, x_4\rangle$ and $\langle y_3, y_4\rangle$ must also be Endgame-consistent. This makes it impossible for $\langle x_1, x_2\rangle$ and $\langle y_1, y_2\rangle$ to be Endgame-inconsistent.

Case 3: *e connects $\langle x_3, x_4\rangle$ in G_1 and $\langle y_3, y_4\rangle$ in G_2 where $\langle x_3, x_4\rangle$ and $\langle y_3, y_4\rangle \in \Phi$(M) and M is N's spouse.* Suppose $\langle u_1, u_2\rangle \in \phi$(M) belongs to the same trio as $\langle x_1, x_2\rangle$ and suppose $\langle v_1, v_2\rangle \in \phi$(M) belongs to the same trio as $\langle y_1, y_2\rangle$. According to the Lemma 5, $\langle u_1, u_2\rangle$ and $\langle v_1, v_2\rangle$ are also Endgame-inconsistent. Thus, we can consider $\langle u_1, u_2\rangle$ and $\langle v_1, v_2\rangle$ instead of $\langle x_1, x_2\rangle$ and $\langle y_1, y_2\rangle$, and accordingly, apply the arguments of Case 2.

Case 4: *e connects $\langle x_3, x_4\rangle$ in G_1 and $\langle y_3, y_4\rangle$ in G_2 where $\langle x_3, x_4\rangle$ and $\langle y_3, y_4\rangle \in \Phi$(M) and M is neither N nor N's spouse.* Assuming no mating loops, this case does not exist. □

2.5 Stage 4

Now we deal with the single connected component G as described before:

Stage 4 – Dealing with Single Component:

1. Arbitrarily pick a vector u of G. For all unresolved locus i, run LOCUS_RESOLVE(G, i, u, 0).
2. For all N, check Φ(N) for Endgame-consistency and report *"no solution"* if it is not maintained.

Theorem 1. *For a given pedigree, we can either achieve a solution that represents a CHC for the given pedigree, or report "no solution" when there is no solution, in $O(mn)$ time where n is the number of nodes in the pedigree and m is the number of loci.* □

3 Concluding Remarks

In this paper, a linear-time algorithm, which is optimal, is presented to solve the haplotype problem for pedigree data when there are no recombinations and the pedigree has no mating loops. We are currently extending the algorithm to handle mating loops.

For the haplotyping problem with recombinations, the problem becomes intractable even when at most one recombination is allowed at each haplotype of a child, or when the problem is to find a feasible haplotype with the minimum number of recombinations (even without mating loops) [4]. However, there is still much scope for further study. For example, in practice, pedigree data often contains a significant amount of missing alleles (up to 14-15% of the alleles belonging to a block could be missing in the pedigree data studied). In some cases, the deduction of the missing information on alleles is possible. The goal is then to devise an efficient algorithm to determine as many missing alleles as possible.

References

1. R. Cox, N. Bouzekri, *et al.* Angiotensin-1-converting enzyme (ACE) plasma concentration is influenced by multiple *ACE*-linked quantitative trait nucleotides. *Hum. Mol. Genet.*, 11:2969-2977, 2002.
2. J. Li and T. Jiang. Efficient rule-based haplotyping algorithms for pedigree data. *RECOMB'03*, pages 197-206, 2003.
3. J. Li and T. Jiang. Efficient inference of haplotypes from genotypes on a pedigree. *J. Bioinfo Comp Biol*, 1(1):41-69, 2003.
4. J. Li and T. Jiang. An exact solution for finding minimum recombinant haplotype configurations on pedigrees with missing data by integer linear programming. *RECOMB'04*, pages 20-29, 2004.
5. J. R. O'Connell. Zero-recombinant haplotyping: applications to fine mapping using SNPs. *Genet Epidemiol*, 19 Suppl 1:S64-70, 2000.
6. E. Russo *et al.* Single nucleotide polymorphism: Big pharmacy hedges its bets. *The Scientist*, 13, 1999.
7. N. Wang, J. M. Akey, K. Zhang, K. Chakraborty, and L. Jin. Distribution of recombination crossovers and the origin of haplotype blocks: The interplay of population history, recombination, and mutation. *Am. J. Hum. Genet.*, 11:1227-1234, 2002.
8. E. M. Wijsman. A deductive method of haplotype analysis in pedigrees. *Am J Hum Genet*, 41(3):356-373, 1987.
9. J. Xiao, L. Liu, L. Xia and T. Jiang. Fast Elimination of Redundant Linear Equations and Reconstruction of Recombination-Free Mendelian Inheritance on a Pedigree. *Manuscript.*