Minimum Parent-Offspring Recombination Haplotype Inference in Pedigrees

Qiangfeng Zhang[1], Francis Y.L. Chin[2], and Hong Shen[3]

[1] Department of Computer Science,
University of Science and Technology of China, Hefei 230026, China
qfzhang@mail.ustc.edu.cn
[2] Department of Computer Science, The University of Hong Kong,
Pokfulam, Hong Kong
chin@cs.hku.hk
[3] Graduate School of Information Science, JAIST, Ishikawa, Japan
shen@jaist.ac.jp

Abstract. The problem of haplotype inference under the Mendelian law of inheritance on pedigree genotype data is studied. The minimum recombination principle states that genetic recombinations are rare and haplotypes with fewer recombinations are more likely to exist. Given genotype data on a pedigree, the problem of Minimum Recombination Haplotype Inference (MRHI) is to find a set of haplotype configurations consistent with the genotype data having the minimum number of recombinations. In this paper, we focus on a variation of the MRHI problem that gives more realistic solutions, namely the k-MRHI problem, which has the additional constraint that the number of recombinations in each parent-offspring pair is at most k. Although the k-MRHI problem is NP-hard even for $k = 1$, the k-MRHI problem with $k > 1$ can be solved efficiently by dynamic programming in $O(nm_0^{3k+1}2^{m_0})$ time by adopting an approach similar to the one used by Doi, Li and Jiang [4] on pedigrees with n nodes and at most m_0 heterozygous loci in each node. In particular, the 1-MRHI problem can be solved in $O(nm_0^4 2^{m_0})$ time. We propose an $O(n^2 m_0)$ algorithm to find a node as the root of the pedigree tree so as to further reduce the time complexity to $O(m_0 min(t_R))$, where t_R is the number of feasible haplotype configuration combinations in all trios in the pedigree tree when R is the root. If the pedigree has few generations, the 1-MRHI problem can be solved in $O(min\{nm_0^4 2^{m_0}, nm_0^{l+1}2^{\mu_R+l}\})$ time, where μ_R is the number of heterozygous loci in the root, and l is the maximum path length from the root to the leaves in the pedigree tree. Experiments on both real and simulated data confirm the out-performance of our algorithm when compared with other popular algorithms. In most real cases, our algorithm gives the same haplotyping results but runs much faster. In some real cases, other algorithms give an answer which has the least number of recombinations, while our algorithm gives a more credible solution and runs faster.

1 Introduction

The modeling of human genetic variation is critical to the understanding of the genetic basis for complex diseases. *Single nucleotide polymorphisms* (*SNPs* [13])

C. Priami, A. Zelikovsky (Eds.): Trans. on Comput. Syst. Biol. II, LNBI 3680, pp. 100–112, 2005.

are the most frequent form of this variation. The Human Genome Project and other large-scale efforts have identified millions of SNP markers that can be used in genetic studies. Although each marker can be analyzed independently, it is much more informative to analyze them in groups. Therefore, it is useful to analyze *haplotypes* (*hap*loid geno*types*), which are sequences of linked markers on a single chromosome. In diploid organisms, such as humans, chromosomes come in pairs, and experiments 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 [2][6].

Input genotype data can be with or without any other *pedigree* information. Haplotyping pedigree data is believed to be more reliable than haplotyping population data for unrelated individuals: the constraint provided by parents-offspring relationships in a pedigree could force one to settle on a unique haplotype configuration as being most probable.

Genetic research shows that recombinations are rare in human data [5]. The genomic DNA can be partitioned into long blocks such that recombinations within each block are rare or even nonexistent. Thus it is believed that haplotype configurations with fewer recombinations should be preferred in haplotype inference [11][12].

The *Minimum-Recombination Haplotype Inference* (*MRHI*) problem, which is NP-hard [4], is to find a haplotype configuration with minimum number of recombinations for a given pedigree genotype data. Various algorithms have been presented for the MRHI problem [8][7][12][15]. In some cases, however, the MRHI model might yield unrealistic results in which a few parent-offspring pairs have many recombinations while others have no or few recombinations. We present a more realistic problem, called the k-MRHI problem which basically is the MRHI problem, but with an additional constraint that the number of recombinations in each parent-offspring pair is bounded by a constant k. The k-MRHI problem is NP-hard even for $k = 1$.

The k-MRHI problem can be solved by a dynamic programming (DP) algorithm which is very similar to the algorithm by Doi, Li and Jiang [4]. By avoiding studying all 2^{3m_0} haplotype configurations in each parents-offspring trio, our algorithm takes $O(nm_0^4 2^{m_0})$ time when $k = 1$, instead of the $O(nm_0 2^{3m_0})$ time needed by [4] for the MRHI problem on pedigrees with n nodes and at most m_0 heterozygous loci in each node. Note that not all nodes have m_0 heterozygous loci, and the number of feasible haplotype configurations at a node is limited by the number of feasible haplotype configurations of its neighbors, and thus the number of possible haplotype configurations at a node can be much less than 2^{m_0}. This observation leads to the idea of choosing different nodes in the pedigree as the root of the tree in speeding up the algorithm. The main contributions of this paper are: (1) to define a more realistic problem for haplotype inference

(k-MRHI), (2) to give a more efficient and practical DP algorithm for the haplotype inference problem with improved time complexities, and (3) to present an efficient algorithm to find the root in the pedigree for better performance in the DP algorithm.

2 Preliminaries

Haplotypes and genotypes consist of linked *genetic markers* which are small segments of DNA with some specific features. 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 two alleles of a biallelic (2-state) SNP can be denoted by '0' and '1', and a haplotype h with m loci is presented as a string of length m over $\{0,1\}^m$, and a genotype g as a string over $\{0,1,2\}^m$. Haplotype pair $\langle h_1, h_2 \rangle$ is compatible with a genotype g if (a) the two alleles of h_1 and h_2 are the same at the same locus, for example '0' (respectively '1'), then the corresponding locus of g should also be '0' (respectively '1'), which denotes a *homozygous* site; otherwise, (b) the two alleles of h_1 and h_2 are different, then the corresponding site of g should be '2', which denotes a *heterozygous* site.

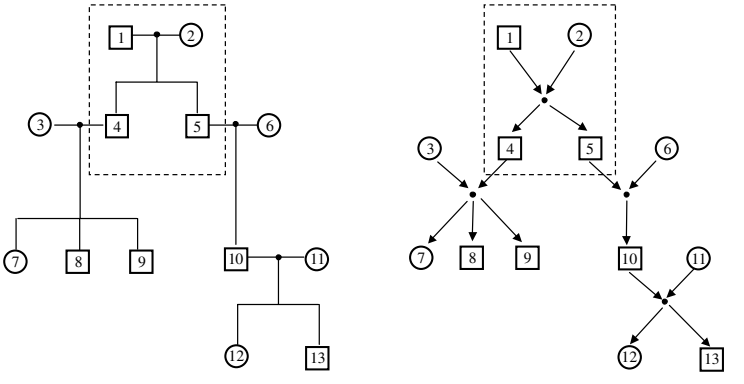

Fig. 1. The pictorial representation and graph representation of a pedigree

A pedigree is a fundamental structure used in genetics. Figure 1(a) shows the pictorial representation (used by the biologists) of a pedigree with 13 nodes. A square represents a male node, a circle represents a female node, and a black dot represents a mating node. The subgraph in the dashed square is a typical nuclear family, which contains a father (node 1), a mother (node 2) and two children (nodes 4 and 5). The children are placed under their parents. Nodes 1, 2 and 4 consist a parents-offspring trio, nodes 1 and 4, nodes 1 and 5, nodes 2

and 4, nodes 2 and 5 are parent-offspring pairs. We define a pedigree formally as in [4].

Defintion 1[4]**.** *A* **pedigree** *is a weakly connected directed acyclic graph* $P = (V, E)$*, where* $V = M \cup F \cup N$*, with* M *stands for the male nodes,* F *the female nodes,* N *the mating nodes, and* $E = \{(u, v)\}$ *with* $u \in M \cup F$ *and* $v \in N$*, alternatively* $u \in N$ *and* $v \in M \cup F$*.*

Figure 1(b) shows the graph representation of the pedigree given in Figure 1(a). A sub-graph containing the father, the mother, and their children is a *nuclear family*. A nuclear family can also be represented by a mating node which connects them together. A *parents-offspring trio*, or just *trio*, consists of two parents and one of their children; and a *parent-offspring pair* (*PO-pair*) refers to a father and his child or a mother and her child. In this paper, we assume that the pedigree never forms a cycle if the directions of edges are ignored (no *mating-loop*).

Each individual node in a pedigree is associated with its genotype. In the absence of genetic mutation, at each locus, the child must inherit one allele from its father and the other from its mother. This is known as the *Mendelian law of inheritance*. Usually, one haplotype of a child is inherited as a whole from one of the two haplotypes of a parent. However, *recombinations* may occur, where the two haplotypes of a parent get shuffled due to a crossover of a chromosome and one of the shuffled copies (*recombinant*) is passed on to the child. However, genetic research shows that recombinations are rare in human genetics. Thus we are interested in finding the haplotype configurations such that the total number of recombinations in the whole pedigree is minimized.

Defintion 2[12]**. Minimum Recombinant Haplotype Inference (MRHI) Problem:** *Given a pedigree graph P, each individual node of P associates with a genotype. Find a haplotype configuration for the pedigree that each haplotype pair at each node is an explanation of its corresponding genoytype and the total number of recombinations is minimized.*

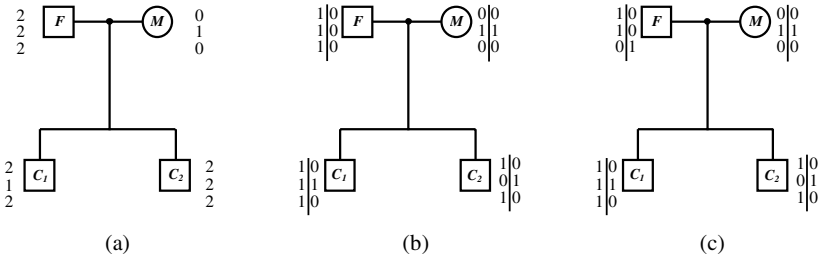

Fig. 2. Two different solutions for a pedigree of a nuclear family

Figure 2(a) shows a pedigree of a nuclear family containing a father F, a mother M and 2 children C_1, C_2. Figure 2(b) gives a solution with no recom-

bination in trio (F, M, C_1) and 2 recombinations in trio (F, M, C_2). Figure 2(c) gives another solution, which also has 2 recombinations in total, but at most one recombination in each trio, i.e. at most one recombination in each PO-pair. As genetic studies show that recombinations are rare to have 2 recombinations within one PO-pair, *e.g.*, there are 13% single recombinations versus 0.84% double recombinations in the Drosophila autosomal genes [5], Figure 2(c) should be a more credible solution than Figure 2(b) for the haplotype inference problem.

Defintion 3 . k-Recombination Haplotype Inference (k-MRHI) Problem: *Given a pedigree graph P with each individual node associated with a genotype, find a haplotype configuration that is compatible with the genotypes at all nodes having the minimum number of recombinations and no more than k recombinations in each PO-pair.*

3 A Dynamic Programming Algorithm for k-MRHI

3.1 The 1-MRHI Problem ($k = 1$)

In [4], Doi *et al.* gives a proof for the NP-hardness of the MRHI problem by a reduction from MAX CUT. In their construction, the number of recombinations within each PO-pair is limited to 1. This trivially implies that the k-MRHI problem, even for $k = 1$, is also NP-hard.

However, in most cases, we can find a feasible solution for a k-MRHI instance with $k < 2$. As we have mentioned before, more than 1 recombination within a PO-pair is very unlikely in reality. Therefore, we shall focus on the 1-MRHI problem first and generalize to the k-MRHI problem later.

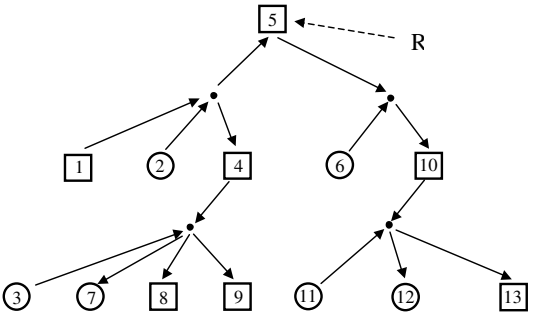

Fig. 3. The searching tree of the pedigree in Figure 1

A locus-based dynamic programming (DP) algorithm for the k-MRHI problem was presented in [4], with a time complexity of $O(nm_0 2^{3m_0})$, where m_0 is the maximum number of heterozygous loci in the genotype at each node of a loopless pedigree. We adopt a similar DP approach to solve the 1-MRHI problem

by (1) assigning an arbitrary node R in the pedigree as the root (an example is given in Figure 3, which shows a rooted tree at node 5 for the pedigree in Figure 1); (2) recursively finding $num[R][s]$, the minimum number of recombinations required for all feasible haplotype configurations s of R; and (3) selecting the haplotype with the minimum number of recombinations as the solution.

Let $num[r][s]$ denote the minimum number of recombinations required in the sub-tree rooted at r with the haplotype configuration s under the constraint that there is at most 1 recombination in each PO-pair of the sub-tree. If r has multiple mating nodes as its tree sons, we compute each mating node separately. Each child mating node of r defines a unique nuclear family, which may contain r as a parent or a child and the computation of $num[r][s]$ is performed recursively in that nuclear family.

Suppose that the nuclear family consists of father F, mother M and children $C_1, \cdots, C_d$. If r is a leaf node, $num[r][s] = 0$ for any of haplotype configuration s; else, if r is M (or F, respectively) with haplotype configuration s, then:

$$num[r][s] = \min_{p}(num[F][p] + \sum_{i=1}^{d} \min_{c_i}(num[C_i][c_i] + numtrio(p, s, c_i))) \quad (1)$$

where p denotes the haplotype configuration at node F and c_i the haplotype configuration at C_i, one of the d children in this nuclear family. numtrio (p, s, c_i) returns the minimum number of recombinations required for a trio consisting of F, M, and C_i with the haplotype configurations p, s and c_i respectively, under the constraint that no PO-pair can have more than one recombination. If there does not have any feasible solution, then numtrio (p, s, c_i) will return ∞, which indicates "no solution".

Similarly, if r is C_j with haplotype configuration s, then we have:

$$num[r][s] = \min_{p,q}(numtrio[p, q, s] + num[F][p] + num[M][q] + \sum_{i=1, i\neq j}^{d} \min_{c_i}(num[C_i][c_i] + numtrio(p, q, c_i))) \quad where\ r = C_j \quad (2)$$

where p, q and c_i are defined as before for haplotype configurations at F, M and C_i respectively.

Note that the above algorithm is the same as that presented in [4], and thus would have the same time complexity. However, a reduction in time complexity is possible from an important observation: it is not necessary to consider all combinations of haplotype configurations in each trio, which number $O(2^{3m_0})$ in total, because many combinations of haplotype configurations will be infeasible, *i.e.* will not have at most one recombination per PO-pair.

For example, assume the genotype of F is $(2, 2, \cdots, 2)$ of length m_0 and with haplotype configuration $s = \langle h_{s1}, h_{s2}\rangle$ and h_{c1} in the haplotype $c = \langle h_{c1}, h_{c2}\rangle$ of C_i is inherited from s with no more than 1 recombination. There are $m_0 + 1$ ways of forming h_{c1} by inheriting its first w alleles from the first w alleles in h_{s1} and the remaining $(m_0 - w)$ alleles from h_{s2} with $0 \leq w \leq m_0$. Similarly, there

are another $m_0 + 1$ ways of forming h_{c1} from the first w alleles in h_{s2} and the remaining $(m_0 - w)$ alleles from h_{s1} . Since there are double-counting in these two cases when $w = 0$ and m_0, the number of feasible haplotype configurations of c is limited to $2m_0$, and the time complexity of the algorithm can be much reduced if we limit the number of configurations needed to be searched for the optimal result. More precisely, suppose r in Equation (1) is M (or F), and $s = \langle h_{s1}, h_{s2} \rangle$, let N_s be the set of feasible haplotype configurations $c = \langle h_{c1}, h_{c2} \rangle$ that can be inherited by child C_i from s of r with no more than one recombination. Thus, $|N_s| \leq 2m_0$. As h_{c2} is inherited from the haplotype configuration $q = \langle h_{q1}, h_{q2} \rangle$ of F, let N_c be the set of feasible haplotype configurations of F which can produce the haplotype configuration c in C with no more than one recombination. Let $N'_{s,C_i} = \cup_{c \in N_s} N_c$, which indicates the set of feasible haplotype configurations at F which can go together with haplotype s at M to produce children C_i with no more than one recombination in the father-child pair and in the mother-child pair. Obviously, $N'_{s,C_i} \leq 4m_0^2$.

As each haplotype configuration of F should be able to produce any of the children $C_1, \cdots, C_d$, the set of feasible haplotype configurations in F is $N'_s = \cap_i N'_{s,C_i}$. Equation (1) can be rewritten as:

$$num[r][s] = \min_{p \in N'_s} (num[F][p] + \sum_i \min_{c_i \in N_s} (num[C_i][c_i] + numtrio(p, s, c_i))) \quad (3)$$

As for Equation (2), if r is C_j and its haplotype configuration $s = \langle h_{s1}, h_{s2} \rangle$, let $N_{s,F}$ and $N_{s,M}$ be the sets of feasible haplotype configurations in F and M, which can produce C_j with haplotype configuration s. As $|N_{s,F}| \leq 2m_0$ and $|N_{s,M}| \leq 2m_0$, let $N_{p,C_i}(N_{q,C_i})$ be the set of feasible haplotype configurations on another child C_i with haplotype configuration p in F (q in M) and $N''_{p,q} = N_{p,C_i} \cap N_{q,C_i}$ be the set of feasible haplotype configurations for each child C_i which can concurrently appear with the haplotype configuration s of child C_j. Note that $N''_{p,q} \leq 2m_0$ and Equation (2) can be rewritten as:

$$num[r][s] = \min_{p \in N_{s,F}, q \in N_{s,M}} (numtrio(p, q, s) + num[F][p] + num[M][q] + \sum_{i \neq j} \min_{c_i \in N''_{p,q}} (num[C_i][c_i] + numtrio(p, q, c_i))) \quad where\ r = C_j \quad (4)$$

Theorem 1. *The above dynamic programming algorithm can solve the 1-MRHI problem in $O(nm_0^4 2^{m_0})$ time and $O(n2^{m_0})$ space for pedigree with n nodes and at most m_0 heterozygous loci in each node.*

Proof. The rooted tree can be constructed in $O(n)$ time. As we have to consider the $8m_0^3$ combinations in each trio for each haplotype configuration of a node and we need $O(m_0)$ time to compute *numtrio* for each haplotype configuration combination in a trio, it may take $O(m_0^4 2^{m_0})$ time to process each trio. There are at most n parent-offspring trios in the pedigree, so the time complexity is $O(nm_0^4 2^{m_0})$. Furthermore, we need to store the array *num* and pointers for backtracking. The size of *num* is $O(n2^{m_0})$, so is the number of pointers. Thus the space complexity is $O(n2^{m_0})$.

3.2 The k-MRHI Problem

We have argued that in most cases, feasible solutions exist for 1-MRHI. However, there are still some instances that require more recombinations within each PO-pair. In almost all practical cases, there are at most 2 recombinations within each PO-pair. In the following, we generalize the DP algorithm to the general k-MRHI problem with some modifications.

We need to modify the definition of neighboring haplotype configurations set from N_s to $N_s^{(k)}$: for each haplotype configuration $c = \langle h_{c1}, h_{c2}\rangle \in N_s^{(k)}$, one of $\langle h_{c1}, h_{c2}\rangle$ is inherited from one of $\langle h_{s1}, h_{s2}\rangle$ with no more than k recombinations. So we have $|N_s^{(k)}| = O(m_0^k)$.

Similarly, we modify the definition of $N_s^{'(k)} = \cap_i N_{s,C_i}^{'(k)}$ with N'_{s,C_i} to $N_{s,C_i}^{'(k)}$ in Equation (3) and the definition of $N_{s,F}$ and $N_{s,M}$ to $N_{s,F}^{(k)}$ and $N_{s,M}^{(k)}$, N_{p,C_i} to $N_{p,C_i}^{(k)}$, and $N''_{p,q}$ to $N_{p,q}^{''(k)}$ in Equation (4). Then we have:

Theorem 2. *The time complexity of the DP algorithm solving the k-MRHI problem is $O(nm_0^{3k+1}2^{m_0})$ for pedigree with n nodes and at most m_0 heterozygous loci in each node.*

4 Root Selection for Better Performance

We have shown in Section 3 that in the 1-MRHI problem, the number of feasible haplotype configuration combinations in each trio is no more than $O(m_0^2 2^{m_0})$. However, in practice the feasible haplotype configuration combinations in each trio may be much less than that because of the following reasons: (1) not all nodes have m_0 heterozygous loci; and (2) the number of feasible haplotype configurations a_v of a node v is also bounded by the number of feasible haplotype configurations a_{v_r} of v's neighbor v_r, which can participate in the feasible haplotype configuration combinations in a trio, *i.e.*, $a_v \leq 2\mu_v a_{v_r}$, where μ_v is the number of heterozygous loci in v. Thus, different selections of a node in the pedigree as the root for the DP algorithm will give different processing times. The following we shall discuss an algorithm to find the best root based on the estimated number of feasible haplotype configurations in each node.

Starting from any node R, as root and assuming α_R be the number of feasible haplotypes configurations of R, *i.e.*, $\alpha_R = 2^{\mu_R}$, we will traverse the tree in pre-order and, for each node v, evaluate the number of the feasible haplotype configurations for its neighboring nodes.

If v has multiple mating nodes as v's children, we compute each mating node separately. Each mating node as v's child defines a unique nuclear family, which may contain v as a parent or a child. Suppose that the nuclear family consists of father F, mother M and children $C_1, \cdots, C_k$.

If v is M (or F, respectively), $\alpha_{C_i} = \min\{2^{\mu_{C_i}}, 2\mu_{C_i}\alpha_v\}$ $(i = 1, \cdots, k)$ and $\alpha_F = \min_i\{2^{\mu_F}, 2\mu_F\alpha_{C_i}\}$. If v is C_i, then $\alpha_F = \min\{2^{\mu_F}, 2\mu_F\alpha_v\}$, $\alpha_M = \min\{2^{\mu_M}, 2\mu_M\alpha_v\}$ and $\alpha_{C_i} = \min\{2^{\mu_{C_i}}, 2\mu_{C_i}\alpha_F, 2\mu_{C_i}\alpha_M\}$ $(i = 1, \cdots, k)$. Thus,

the number of feasible haplotype configuration combinations t_i in trio T_i can be computed consequently, assuming an arbitrary node (node R) as the root of the searching tree. The total number of feasible haplotype configuration combinations in all trios in the pedigree is $t_R = \sum_i t_i$, which can be computed by a tree traversal.

Theorem 3. *Let m_0 be the number of heterozygous loci and t_R be the total number of feasible haplotype configuration combinations for all trios in the pedigree with node R as root. Then the node which gives $\min(t_R)$ can be found in $O(n^2 m_0)$ time and the 1-MRHI problem can be solved in $O(m_0 \min(t_R))$ time.*

Proof. We can evaluate t_R for each node R in the pedigree in $O(nm_0)$ time and choose the node with $\min(t_R)$ as the root in $O(n^2 m_0)$ time. As the computation of numtrio for each feasible haplotype configuration combination in each trio takes $O(m_0)$ time, the 1-MRHI problem can be solved in $O(m_0 \min(t_R))$ time after selecting the best root for the DP algorithm.

4.1 Special Pedigrees with Few Generations

We notice that the diameters of the pedigree graphs in many practical instances are usually small. For example, the 452 families in the CEPH database [1][3][10] consist of only three generations, usually with four grandparents, two parents and a number of children. Figure 4 shows a typical family (family 1413) with 21 members. The longest path starts from one of the grandparents from the father's side to one of the grandparents from the mother's side and is of length 4 (not counting the mating nodes). But if we start from any of the children, we can reach any other node within 2 steps.

Suppose that the number of heterozygous loci in the chosen root R is μ_R, and any other nodes can be reached within l steps from R. We shall enumerate all the $2\mu_R$ feasible haplotype configurations of the root in the first step, and no more than $2^{\mu_R} \times 2m_0$ feasible haplotype configurations for each of its neighboring nodes in the second step, and so on. The number of feasible haplotype configurations

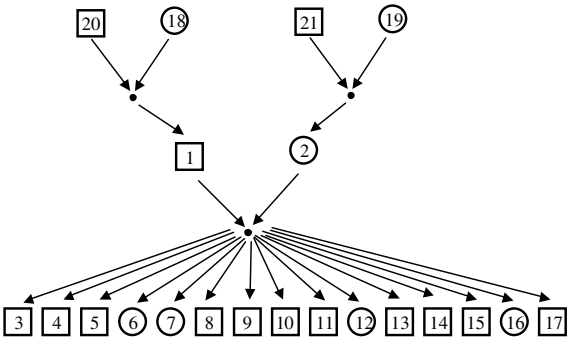

Fig. 4. A typical family(family 1413) in the CEPH database

is at most $2^{\mu_R} \times (2m_0)^l$ at the most distant node. When $\mu_R \ll m_0$ and l is relatively small, we will get an improvement in the time complexity:

Theorem 4. *1-MRHI can be solved in* $\min\{O(nm_0^4 2^{m_0}), O(nm_0^{l+1} 2^{\mu_R+l})\}$ *time for pedigree with n nodes and at most* m_0 *heterozygous loci in each node, where* l *is the maximum path length from the root to the leaves and* μ_R *is the number of heterozygous loci in root* R.

Proof. We have to consider all combinations of feasible haplotype configurations at nodes in each trio, which is at most $2^{\mu_R} \times (2m_0)^l$. We need $O(m_0)$ time to compute *numtrio* for each haplotype configuration combination in each trio, and there may be at most $O(n)$ trios, the time complexity of the algorithm is $\min\{O(nm_0^4 2^{m_0}), O(nm_0^{l+1} 2^{\mu_R+l})\}$.

5 Experimental Results

We implemented the above DP algorithm in C++, and all experiments were conducted on a Pentium IV PC with 1.7GHz CPU and 256MB RAM.

5.1 Real Data

We examined a real data set on Epsiodic Ataxia (EA) by Litt *et al.*[9] which involves a family containing 29 people typed at 9 polymorphic markers on chromosome 12p. Both the locus-based algorithm [4] and the 1-MRHI algorithm run fast ($t < 1$ *sec.*) on this data set but the results are different. The locus-based algorithm gives a feasible solution with 5 recombinations in total but with a double recombination in one haplotype of member 100. The 1-MRHI algorithm, however, finds a more credible solution that has 6 recombinations in total, but with at most 1 recombination for each haplotype in the pedigree.

Another two real data sets are three generations families like those in the CEPH database [1][3][10] (*ftp://genome.wi.mit.edu/distribution/mpg/hapmap/hap_struct/popA/* (*Gabriel et al.*)): family 1331 on chromosome 7a, and family 1346 on chromosome 2a. After removing the loci with missing alleles, family 1331 is a pedigree consisting of 8 members on 32 loci, and family 1346 is a pedigree consisting of 8 members on 55 loci. Both the locus-based algorithm and the 1-MRHI algorithm give the same answer for family 1331, but take 522.4s and 8.7s, respectively. As for family 1346, the locus-based algorithm fails because of not enough resources while the 1-MRHI algorithm finds out a solution in 31 minutes.

5.2 Simulated Data

We compare the performance of our algorithm, with the locus-based algorithm [4] and PHASE [14], a widely used program based on Gibbs Sampling algorithm, the *running time* t and the *accuracy ratio* ρ (in recovering the correct haplotypes). We used three different tree pedigree structures in the experiment: (1) a tree

with 13 nodes (Figure 1), (2) a tree with 29 nodes (Figure 8 in [7]), (3) a typical family with 21 nodes from the CEPH database [1][3][10] (Figure 5).

For each pedigree, genotypes with 15 and 30 biallelic marker loci are considered. The two alleles at each locus of a founder are independently sampled with a fixed frequency of 0.5. The recombination rate is set to $r = 0$, 0.1, 0.2 between generations, and we limit the number of recombinations to no more than one in each PO-pair. For each combination of the above parameters, 100 sets of random genotype data are generated and the average performance of the programs is computed, as shown in Table 1.

Table 1. Comparison of performances of different haplotyping programs on simulation data

	$m = 15$						$m = 30$	
	locus-based[4]		PHASE[14]		1-MRHI		1-MRHI	
(n, r)	$t(sec.)$	ρ	$t(sec.)$	ρ	$t(sec.)$	ρ	$t(sec.)$	ρ
(13, 0.0)	255.7	1.00	688.2	.87	1.68	1.00	202.8	1.00
(29, 0.0)	576.3	1.00	1772.8	.91	12.33	1.00	839.6	1.00
(21, 0.0)	234.4	1.00	592.4	.95	1.02	1.00	44.0	1.00
(13, 0.1)	287.7	.93	972.3	.85	1.73	.91	241.1	.92
(29, 0.1)	542.8	.90	2210.2	.90	10.45	.90	1042.8	.94
(21, 0.1)	243.2	.91	1504.2	.93	0.52	.94	33.7	.96
(13, 0.2)	294.2	.85	1221.4	.85	3.17	.89	1032.4	.86
(29, 0.2)	613.5	.81	3022.2	.89	11.70	.84	916.1	.84
(21, 0.2)	244.1	.90	2106.7	.93	1.22	.95	47.4	.92

† Average performance is obtained from 100 independent executions of each program and for each parameter setting. n stands for the number of nodes, m for the number of marker loci, r for the recombination rate, $t(sec.)$ for the average running time, and ρ for the accuracy ratio.

‡ The locus-based algorithm cannot be applied to cases of $m \geq 30$, due to the space limitation. PHASE is also excluded for cases of $m \geq 30$ because of the time.

As we can see from the table, 1-MRHI runs quickest, and the locus-based algorithm runs quicker than PHASE. Thus the 1-MRHI algorithm can be applied to much larger instances than the locus-based algorithm and PHASE can (the other two algorithms fail when $m = 30$).

In terms of the quality of solutions, all three algorithms can recover the correct haplotype configurations with high probabilities. The accuracy ratio decreases with the increase in the number of recombinations, which is more obvious for the locus-based algorithm and the 1-MRHI algorithm. Since we have limited the number of recombinations within each PO-pair to no more than 1 in the data, the locus-based algorithm, which often finds solutions with fewer recombinations than the actual haplotype configurations, performs worse than the 1-MRHI algorithm as expected.

6 Concluding Remarks

1-MRHI brings an improvement on the running time of solving the general MRHI problem, even though 1-MRHI and the general MRHI usually give the same solutions as confirmed from the experiments. If the solutions are different, 1-MRHI usually gives the more credible solutions. In some cases, if the total number of recombinations for 1-MRHI solutions is much larger than the total number of recombinations for 2-MRHI solutions, then it is plausible that the 2-MRHI solution should be more credible. Our next goal is to find the most probable haplotype configuration which can explain the genotypes in a pedigree when the probabilities of single, double, triple recombinations are given.

Our algorithm for k-MRHI cannot deal with mating loops; nor can the locus-based DP algorithm [4]. A member-based DP algorithm [4] can deal with pedigrees with mating loops, but may not be well-suited to solving the k-MRHI problem because of the increase in time complexity. In practice, pedigree data often contains missing alleles. It will be interesting to find an efficient algorithm for k-MRHI on pedigrees with mating loop and genotypes with missing data.

Acknowledgement. We thank T. Jiang and J. Li for sharing their DP code with us and Dr. M.Y. Chan for proof-reading the first draft of this paper. Thanks to the RGC grant HKU 7135/04E for supporting this research.

References

1. *The CEPH genotype database.* http://www.cephb.fr/.
2. A.G. Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. *Mol. Biol. Evol*, 7(2):111–122, 1990.
3. J. Dausset, H. Cann, D. Cohen, M. Lathrop, J.M. Lalouel, and R. White. Centre d'etude du polymorphisme humain (ceph): collaborative genetic mapping of the human genome. *Genomics*, 5:575–577, 1990.
4. K. Doi, J. Li, and T. Jiang. Minimum recombinant haplotype configuration on tree pedigree. In *Proc. of WABI'03*, pages 339–353, 2003.
5. A. Griffiths, W. Gelbart, R. Lewontin, and J. Miller. *Modern Genetic Analysis: Integrating Genes and Genomes.* W.H. Freeman and Company, N.Y., 2002.
6. D. Gusfield. Inference of haplotypes from samples of diploid populations: complexity and algorithms. *J. Computational Biology*, 8:305–323, 2001.
7. J. Li and T. Jiang. Efficient inference of haplotypes from genotypes on a pedigree. *J. Bioinfo Comp Biol*, 1(1), 2003.
8. Jing Li and Tao Jiang. Efficient inference of haplotypes from genotypes on a pedigree. In *Proc. of RECOMB'03*, pages 197–206, 2003.
9. M. Litt, P. Kramer, D. Browne, S. Gancher, E.R.P. Brunt, D. Root, et al. A gene for episodic ataxia/myokymia maps to chromosome 12p13. *Am. J. Hum. Genet.*, 55:702–709, 1994.
10. J.C. Murray et al. A comprehensive human linkage map with centimorgan density. *Science*, 265:2049–2054, 1994.
11. J.R. O'Connell. Zero-recombinant haplotyping: applications to fine mapping using snps. *Genet Epidemiol*, 19, 2000.

12. D. Qian and L. Beckmann. Minimum-recombinant haplotyping in pedigrees. *Am J Hum Genet*, 70(6):1434–1445, 2002.
13. E. Russo et al. Single nucleotide polymorphism: Big pharma hedges its bets. *The Scientist*, 13, 1999.
14. M. Stephens, N.J. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction for population data. *Am. J. Hum. Genet*, 68:978–989, 2001.
15. P. Tapadar, S. Ghosh, and P.P. Majumder. Haplotyping in pedigrees via a genetic algorithm. *Hum Hered*, 50(1):43–56, 2000.