Haplotype Inference in Complex Pedigrees

Bonnie Kirkpatrick[1], Javier Rosa[2], and Eran Halperin[3], Richard M. Karp[4]

[1] Computer Science Dept, University of California, Berkeley. Email: bbkirk@eecs.berkeley.edu
[2] Computer Science Dept, Rutgers, The State University of New Jersey, New Brunswick
[3] International Computer Science Institute, Berkeley. Email: heran@icsi.berkeley.edu
[4] Computer Science Dept, University of California, Berkeley and International Computer Science Institute. Email: karp@icsi.berkeley.edu

Abstract. Despite the desirable information contained in complex pedigree datasets, analysis methods struggle to efficiently process these datasets. The attractiveness of pedigree data sets is their power for detecting rare variants, particularly in comparison with studies of unrelated individuals. In addition, rather than assuming individuals in a study are unrelated, knowledge of their relationships can avoid spurious results due to confounding population structure effects. However, a major challenge for the applicability of pedigree methods is the ability handle complex pedigrees, having multiple founding lineages, inbreeding, and half-sibling relationships.
A key ingredient in association studies is imputation and inference of haplotypes from genotype data. Existing haplotype inference methods either do not efficiently scales to complex pedigrees or their accuracy is limited. In this paper, we present algorithms for efficient haplotype inference and imputation in complex pedigrees. Our method, PhyloPed, leverages the perfect phylogeny model, resulting in an efficient method with high accuracy. In addition, PhyloPed effectively combines the founder haplotype information from different lineages and is immune to inaccuracies in prior information about the founders.

1 Supplementary Materials

1.1 Updating Lineage Haplotype Assignments

We describe how the haplotype assignments in a single lineage are updated within one iteration of our algorithm. Assume that we have a consistent, non-recombinant state for the pedigree (obtained by the method in 1.2). For the moment, consider restricted instances of a general lineage $L(p, q)$ for founders p and q. Further, consider the subgraph of $L(p, q)$ that contains only one child, r, of MFP (p, q). Call this subgraph L_r. For these restricted instances we assume:

1. L_r is a tree; *i.e.*, there is no inbreeding. This means that except for r, who has two lineage haplotypes, each node in L_r has one non-lineage parent $n(r)$ and one lineage parent $l(r)$ in L_r. If a lineage individuals has two non-inbred parents in the lineage, flip a coin to choose which parent will be the non-lineage parent for the current update step.
2. We assume that each non-lineage founder is the parent of at most one node in L_r.
3. For each non-lineage parent v, the haplotype that v transmits to its child in L_r is drawn from a given prior probability distribution α, and the haplotype transmissions from the different non-lineage parents are mutually independent. For a non-lineage founder, all haplotypes are considered transmittable. However, for non-founder, non-lineage parents, the set of transmittable haplotypes is limited to $\{h_{n(r)}, h'_{n(r)}\}$ and must be from a consistent, non-recombinant state of the pedigree

Our algorithm is similar to the peeling plus random propagation scheme described in [3] and has two steps, the dynamic programing step moving up the lineage and the random propagation of new haplotype assignments down the lineage.

Dynamic Programming. Let indices i and j denote haplotypes. For each node w in L_r let $T(w)$ be the subtree consisting of w and all its descendants; in particular, $T(r) = L_r$. If $w \neq r$ let $a(w)$ denote the haplotype that w receives from its non-lineage parent, and $b(w)$, the haplotype that w receives from its lineage parent in $L(p, q)$, and let $\alpha(i)$ be the probability that $a(w) = i$. Let $a(r)$ and $b(r)$ the two lineage haplotypes of r (all relevant properties are unaffected by interchanging $a(r)$ and $b(r)$).

Recall that $C(w,i,j)$ is an indicator variable that is 1 if the haplotype pair (i,j) is consistent with some genotype in $S(w)$. Recall also that $H(w)$ denotes the set of children of w. We want to compute:

$$\phi_r(i,j) = \sum_{b(x),a(x)\forall x\in T(r)} C(r,i,j)Pr[a(r)=i,b(r)=j|\text{MFP haplotypes}] \cdot \prod_{x\in T(r)} Pr[b(x),a(x)|b(l(x)),a(l(x)),h_{n(x)}),h_{n(x)})]$$

If we re-arrange the equation, we get a message for each individual, that is a factor in the complete marginal. Then for leaf-nodes u, and internal nodes w, where $w \neq r$ we have the recursive equations:

Leaf: $\phi_u(i) = \sum_j \alpha(j)C(u,i,j)$

Internal Node: $\phi_w(i) = \sum_j \alpha(j)C(w,i,j)\prod_{x\in H(w)} \frac{\phi_x(i)+\phi_x(j)}{2}$

Founder Child: $\phi_r(i,j) = C(r,i,j)\prod_{x\in H(r)} \frac{\phi_x(i)+\phi_x(j)}{2}$

We can compute all these quantities by working upward from the leaves of the tree L_r to the root. The running time of the algorithm is $O(N^2V)$ where N is the number of haplotypes and V is the number of individuals in L_r.

We now extend the restricted model to encompass the monogamous pair of founders, p and q in their lineage $L(p,q)$. Extending the restricted model to this case, we make the same assumptions as above except that the MFP-children $r \in H(p)$ have two lineage haplotypes. We continue to assume that each non-lineage parent of a child in $L(p,q)$ is the parent of at most one node in $L(p,q)$ and that a probability distribution α independently determines the probability of haplotype transmission for each non-lineage haplotype.

Let $\phi_{p,q}(i,j,k,l)$ be the probability that the haplotypes of all nodes of $L(p,q)$ are compatible with the genotype data at those nodes, given that p has haplotypes i and j and q has haplotypes k and l. Then the recursive probabilities for the **founder pair** are

$\phi_{p,q}(i,j,k,l) = C(p,i,j)C(q,k,l)\prod_{r\in H(p)} \frac{\phi_r(i,k)+\phi_r(i,l)+\phi_r(j,k)+\phi_r(j,l)}{4}$

where, for each $r \in H(p)$, the quantities on the right-hand side are computed according to the recursive algorithm given above. The execution time of the computation is $O(N^4V)$.

Random Propagation. Once we have computed the quantities $\phi_{p,q}(i,j,k,l)$, $\phi_r(i,j)$ and $\phi_w(i)$ for the MFP (p,q) and their descendants we can sample efficiently from the conditional distribution of the joint assignments of haplotype pairs to all nodes of $L(p,q)$, subject to the requirement that the haplotype assignment at each node is compatible with a consistent, non-recombinant state for all their descendants. Each sample is computed by traversing $L(p,q)$ top-down, starting at the source nodes p and q. The haplotype pair assigned to each node is determined after the haplotype pairs assigned to its parents have been determined.

In detail, the process is as follows. As before, let i,j denote the two haplotypes of p and let k,l the two haplotypes of q. The probability that, in the sample, that the haplotypes of p and q are set to the particular values (i,j,k,l) is proportional to $\alpha(i)\alpha(j)\alpha(k)\alpha(l)\phi_{p,q}(i,j,k,l)$, where the constant of proportionality makes the probabilities of all choices sum to 1. Similarly, let r be a child of the founding pair. Then the probability that r's haplotypes are set to (i,k) is proportional to $\phi_r(i,k)$, and a similar formula holds for the three other possible choices (i,l), (j,k), and (j,l). If w is a node of $L(p,q)$ whose parent z is not p,q, or r then the sampling process sets the lineage haplotype as either h_z or h'_z with probability proportional to $\phi_w(h_z)$ and $\phi_w(h'_z)$. The probabilities of the other joint choices are computed similarly. The execution time of the sampling process is $O(N^4V)$.

We now extend our algorithm to deal with violations of the tree-like model. The violations are of two types:

1. A non-lineage founder may be a parent of more than one node among the descendants of the monogamous founding pairs.
2. A node may be the child of two nodes descended from the same MFP. In that case, one of the parents is arbitrarily designated as the non-lineage parent and the edges from such non-lineage parents are excluded in forming the dags $L(p,q)$ descending from the monogamous founding pairs. As a result, each descendant of a monogamous founding pair lies in exactly one of these dags.

1.2 Branch-and-Bound for Non-recombinant Haplotype Inference

This algorithm initializes the algorithm in the previous section. Here we describe how to obtain a consistent, non-recombinant pedigree state (with a pair of haplotypes for every individual). We use a branch-and-bound algorithm that branches on possible haplotype states for each nuclear family and bounds consideration of haplotypes states when those haplotypes would yield a lower transmission likelihood than a previously observed pedigree state. The transmission likelihood for a pedigree state $\{s_w | w \in I\}$ is defined as:

$$\prod_{w \in I \setminus F} C(w, s_w) Pr[s_w | s_{f(w)}, s_{m(w)}].$$

Preprocessing. The preprocessing step is a linear-time operation that reduces the number of branching options that our algorithm will take. Assume the pedigree is ordered topologically. This means that the first node considered will be an individual without children and that children must be considered before their parents. Each nuclear family is considered in topological order until reaching the founders. Each trio is quickly considered to eliminate haplotype pairs for each individual that are inconsistent with the genotypes of other trio members. Recall that $S(w)$ is defined as set of haplotype pairs that are consistent with the observed genotypes of individual w. Typically $|S(w)| = 2^{N_w}$ where N_w is the number of heterozygous loci in w's genotype. However during the preprocessing step means that we effectively reduce the size of $S(w)$ by removing haplotype pairs that are inconsistent with the genotypes of w's nuclear family.
Alternatively, we can choose to restrict our attention to the subsets $P(w) \subset S(w)$ having the property that $H = \{P(w) | w \in I\}$ are haplotypes compatible with a single perfect phylogeny tree. For haplotypes this is equivalent to the stipulation that the evolution of the founding haplotypes occurred without recombination according to the infinite-sites coalescent model [2]. We use the *BuildTree* algorithm in [1] to quickly find the $o(2^{M-1})$ perfect phylogeny trees compatible with the genotyped individuals, where M is the number of SNPs.

Branching. For each nuclear family (in topological order), the algorithm branches on each possible haplotype assignment to the two parents. We get this list of assignments by listing the possible assignments to both parents that are consistent with the genotypes of the children. Once a haplotype assignment is made to both the parents, the children are assigned haplotypes (if they were not already assigned when considering a previous nuclear family). The haplotypes for the children are chosen to maximize the transmission likelihood conditional on the assignments for the parents.
In order to get the consistent, non-recombinant haplotype-pairs for the parents, we use boolean logic. For the first parent, we make a boolean expression that yields all pairs of haplotypes that are consistent with the genotypes of all the children and one of the parents (irrespective of the other parent's genotype). This list contains only haplotype pairs that maintain Mendelian consistency with the children. Once we have the list of feasible haplotype assignments for this parent, we branch the computation and make a different assignment to the first parent in each branch.
The second parent's haplotypes are assigned conditional on the assignment already made to the first parent, and there is a similar boolean expression that gives a list of haplotype-pairs that are now consistent with all the individuals in the nuclear family. Again, the algorithm branches once for each distinct haplotype assignment for the second parent.

Bounding. The branching process proceeds up the pedigree either until all individuals are assigned haplotypes, or until a bounding opportunity appears. For each individual encountered while proceeding up the pedigree, the algorithm updates the partial log of the transmission likelihood which contains terms for individuals appearing later in the topological ordering. If the partial log-likelihood becomes smaller than the log-likelihood for a previously observed state, then the algorithm ceases to follow that branch of the computation.

1.3 Supplementary Results

Accuracy. In the paper we gave the average accuracy of each method on all of the blocks, including blocks which violated the assumptions of no recombination. In Table 1 we show that when considering only blocks compatible with the perfect phylogeny, all the methods perform slightly better, but

PhyloPed has a greater advantage. Intuitively this is because PhyloPed has a greater advantage in situations that satisfy the assumptions of our method.
Figure 1 shows the box-plots for the data given in Fig. 1 of the paper.These notched box-plots show that the medians of the methods are significantly different–meaning that the means are significantly unlikely to be the same. Other simulations produced similar plots (data not shown).

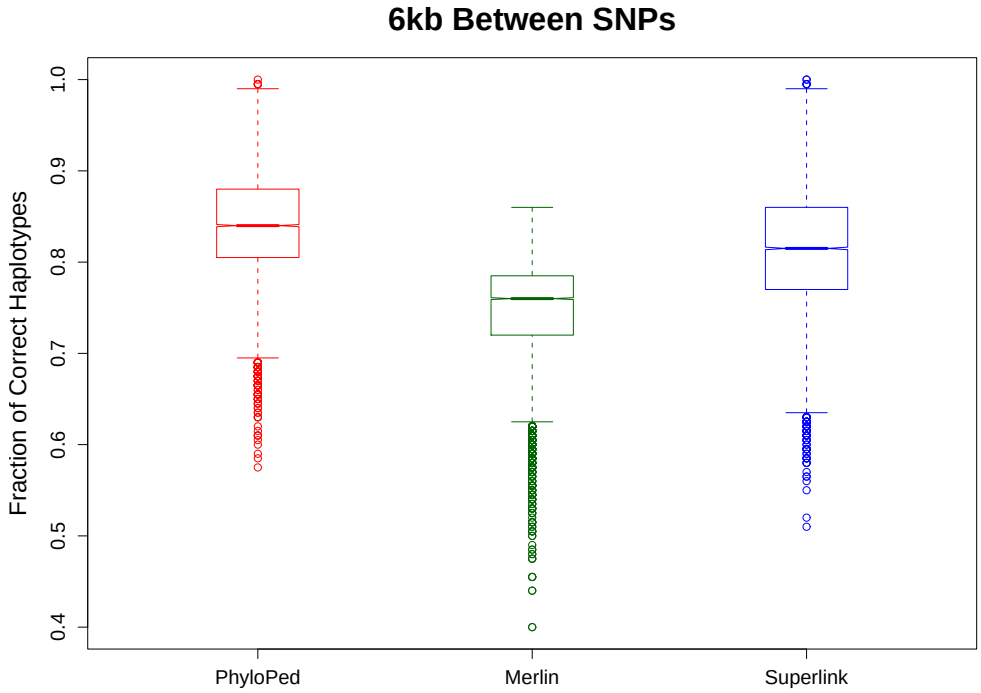
		All blocks		**Perfect Phylo. Blocks**	
Pedigree	**Method**	**Avg**	**Std-Dev**	**Avg**	**Std-Dev**
L1	PhyloPed	**0.867**	0.030	**0.888**	0.018
	Merlin	0.855	0.018	0.861	0.018
	Superlink	0.836	0.034	0.840	0.036
S1	PhyloPed	**0.809**	0.065	**0.825**	0.061
	Merlin	-	-	-	-
	Superlink	0.796	0.064	0.803	0.067
M1	PhyloPed	**0.808**	0.060	**0.835**	0.060
	Merlin	-	-	-	-
	Superlink	0.795	0.058	0.815	0.051
H1	PhyloPed	**0.816**	0.161	**0.821**	0.164
	Merlin	0.750	0.138	0.752	0.140
	Superlink	0.795	0.129	0.795	0.134

Table 1. Average accuracy and standard deviation. When scoring accuracy only on blocks where the genotypes are compatible with the perfect phylogeny, PhyloPed performs even better relative to the other methods. The dash indicates that Merlin was unable to execute due to running time.

Convergence. PhyloPed uses the l_1 deviation of the α^t estimates as a convergence criteria. Figure 2 shows that this deviation drops rapidly. Most of the the blocks in Fig. 1 converged in roughly 6-8 iterations (data not shown).

References

1. E. Eskin, E. Halperin, and R. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. *Journal of Bioinformatics and Computational Biology*, 1(1):1–20, 2003.
2. Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. In *Proceedings of the 6th Annual International Conference on (Research in) Computational (Molecular) Biology*, 2002.
3. S. L. Lauritzen and N. A. Sheehan. Graphical models for genetic analysis. *Statistical Science*, 18(4):489–514, 2003.



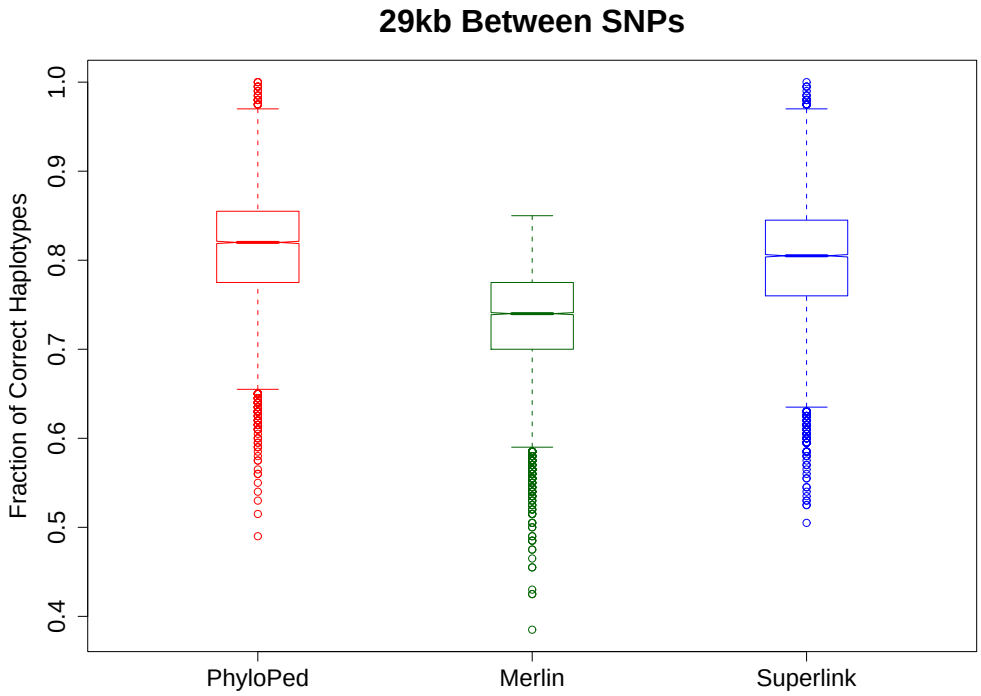


Fig. 1. Accuracy Against Recombination Rate. This plot shows results of 10000 blocks for the 2-lineage, 10-individual family. The accuracy of each method was computed for different recombination rates between neighboring SNPs.

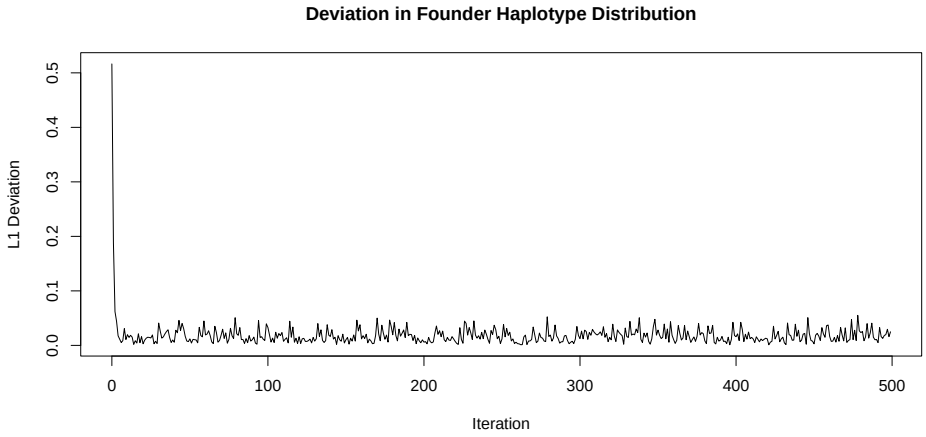


Fig. 2. Convergence of the l_1 deviation between consecutive estimates for the founder haplotype distribution.