TOWARD AN ALGEBRAIC UNDERSTANDING OF HAPLOTYPE INFERENCE BY PURE PARSIMONY

Daniel G. Brown and Ian M. Harrower

*David R. Cheriton School of Computer Science, University of Waterloo,
200 University Avenue W.,
Waterloo, Ontario, Canada N2L 3G1
Email: {browndg,imharrow}@cs.uwaterloo.ca*

Haplotype inference by pure parsimony (HIPP) is known to be NP-Hard. Despite this, many algorithms successfully solve HIPP instances on simulated and real data. In this paper, we explore the connection between algebraic rank and the HIPP problem, to help identify easy and hard instances of the problem. The rank of the input matrix is known to be a lower bound on the size an optimal HIPP solution. We show that this bound is almost surely tight for data generated by randomly pairing p haplotypes derived from a perfect phylogeny when the number of distinct population members is more than $(\frac{1+\epsilon}{2})p \log p$ (for some positive ϵ). Moreover, with only a constant multiple more population members, and a common mutation, we can almost surely recover an optimal set of haplotypes in polynomial time. We examine the algebraic effect of allowing recombination, and bound the effect recombination has on rank. In the process, we prove a stronger version of the standard haplotype lower bound. We also give a complete classification of the rank of a haplotype matrix derived from a galled tree. This classification identifies a set of problem instances with recombination when the rank lower bound is also tight for the HIPP problem.

Keywords. Haplotype inference, phylogenetic networks, galled tree, probabilistic analysis of algorithms

1. INTRODUCTION

Haplotype inference is the process of attempting to identify the chromosomal sequences that have given rise to a diploid population. Recently, this problem has become increasingly important, as researchers attempt to connect variations to inherited diseases. The simplest haplotype inference problem to describe is *haplotype inference by pure parsimony* (HIPP), introduced by Gusfield[1]. The goal is to identify a smallest set of haplotypes to explain a set of genotypes. This objective is partly justified by the observation, now made in several species[2, 3], that in genomic regions under strong linkage disequilibrium, few ancestral haplotypes are found.

The problem is also interesting purely combinatorially; it is NP-hard[1], and its only known polynomial-time approximation algorithms have exponentially bad performance guarantees[4]. In practice, though, it is surprisingly easy to solve, especially when applied to synthetically generated test instances arising from standard evolution models. Several authors have developed integer programming or branch-and-bound algorithms that have shown fast performance[1, 5–7], and some real-world problems have readily solved exactly or near-exactly[7].

To help resolve this conflict between theoretical complexity and practical ease of solution, we explore the algebraic structure of the problem. We focus on problem instances resulting from random pairing of haplotypes generated by standard evolution models. Our results focus on the algebraic rank of the genotype matrix. This is known to be a lower bound to the optimal solution size of the HIPP problem[8]. We show this bound is almost surely tight for data generated by $(\frac{1+\epsilon}{2})p \log p$ random pairings of p haplotypes from a perfect phylogeny (for positive ϵ).

In Section 5, we strengthen our results by exploring population models that allow for recombination. Here, we show that for haplotypes with few recombinations, the rank bound is near-exact, and we show in particular the conditions under which it *is* exact for galled trees, the simplest kind of phylogenetic networks that include recombination. In addition, this study of algebraic rank allows us to prove a new variation of the “haplotype lower bound”[9] on the number of recombinations required to explain a haplotype matrix.

The rank lower bound is important: when tight, it verifies for branch-and-bound procedures that they have found an optimum; moreover, computing the *size* of the HIPP solution is NP-hard in general. However, we can go one step further for some HIPP instances, to compute an optimal set of haplotypes in polynomial time. Our second theorem, in Section 4, shows a constructive algorithm that almost surely runs in polynomial time on instances derived from perfect phylogenies with a constant factor more genotypes than for our first theorem. Our result requires that there be a mutation found in at least an α fraction of the genotypes. We connect this theorem to the standard coalescent model from population genetics in Section 4.1, where we show that for haplotypes generated by this model, we can guarantee such a common mutation exists with probability at least $1 - \epsilon$ when the mutation parameter θ of the coalescent model is at least a constant depending on ϵ and α.

Our results identify structure in HIPP instances and show how to easily solve HIPP in such cases, despite its theoretical hardness. Although we do not show why previous haplotype inference algorithms often run surprisingly quickly, we do show that many of the instances upon which they are run have special structure, which reduces the complexity of the problem.

2. BACKGROUND AND RELATED WORK

We begin by briefly reviewing existing work on haplotype inference, particularly with the parsimony objective, and on a phase transition for haplotyping problems.

2.1. Haplotype inference and notation

The input to a haplotype inference algorithm is a *genotype matrix*, G. Each of its n rows represents the genotype g_i of population member p_i; the m columns represent sites in the genome. $G(i, j)$ is the genotype of population member p_i at site s_j.

We assume there are two alleles, 0 and 1, at each site, and thus three choices for $G(i, j)$: $G(i, j) = 0$ if both parent chromosomes of p_i have allele 0 at s_j, $G(i, j) = 1$ at positions where one parent has each allele, and $G(i, j) = 2$ if both have allele 1. (**Note**: this is **not** the standard notation, which exchanges the meanings of 1 and 2.)

Haplotype inference consists of explaining G with a 0/1 *haplotype matrix*, H. The k rows of H represent possible chromosome choices and their alleles at the m sites. Genotype g_i is explained by H if there exist two rows (possibly the same) of H whose sum is g_i. We can represent this pairing by an $n \times k$ *pairing matrix*, Π, where row r_i of Π has value 1 in the two columns corresponding to the two parent haplotypes of g_i, and 0 elsewhere. (If two copies of the same haplotype explain genotype g_i, the row of Π has a 2 in that column and 0 elsewhere.) In our formulation, haplotype inference consists of finding a haplotype matrix H and a valid pairing matrix Π such that $G = \Pi \cdot H$. (The simplicity of this formulation explains our notational choice; it is inspired by a formulation of He and Zelikovsky[10].)

2.2. Pure parsimony

In *haplotype inference by pure parsimony* (HIPP), introduced by Gusfield[1], we seek the smallest set of haplotypes to explain G. This corresponds to finding the smallest H such that a proper pairing matrix Π exists where $G = \Pi \cdot H$. It is NP-hard[1], and its known approximation algorithms have only exponential approximation guarantees[4].

Still, many instances of this problem have been easy to solve. Gusfield[1] gave an integer linear programming formulation for the problem that, though theoretically exponential in size, often solved very quickly. Halldórsson *et al.* found a polynomial-sized IP formulation for the problem[11]; we independently identified and extended it with further inequalities[5, 7]. Our experiments demonstrated that even on large instances, the optimal solution can be found easily. Why are HIPP instances often solvable in practice?

2.3. Haplotypes and genotypes

To answer this question, we examine some models for generating HIPP instances: where do haplotypes come from, and how are they paired to make genotypes?

The simplest generative model for haplotypes is perfect phylogeny. In this model, all m sites evolve according to a rooted phylogenetic tree, with the root at the top of the tree. Without loss of generality, we assume that at every sampled site s_j, the common ancestor of all haplotypes had allele 0. Each site is assigned to a single edge of the tree, which represents when the unique mutation of that site occurred. Leaves descendant from that point have allele 1 at site s_j; other leaves have allele 0. A matrix H compatible with this framework is a PPH matrix (for *perfect phylogeny haplotype*; we can also relax the all-zero ancestor requirement). For ease of calculation, we also include a single non-polymorphic site s_0, where all haplotypes have value 1; this has no effect on any solution, since any haplotype pairing gives the same genotype, 2, at that site.

More complex generative models for haplotypes allow for recombination, breaking the rule that every site derives from the same single phylogenetic tree. We discuss recombination in Section 5. We can also allow a probabilistic process for generation of the haplotype matrix; the simplest of these is described in Section 4.1, where we consider model parameter settings that make the conditions of our second major theorem (Theorem 4.1) likely to hold.

Haplotypes pair to form genotypes. The simplest process is random pairing: each genotype results from pairing two haplotypes sampled with replacement. This corresponds to n edges being picked from a random multigraph model; every edge (i, j) has probability $2/k^2$, and every loop (i, i) has probability $1/k^2$. A haplotype or genotype may occur in multiple copies.

2.4. A phase transition for haplotyping?

In 2003, Chung and Gusfield[12] examined the number of distinct PPH solutions to a genotype matrix obtained by randomly pairing $2n$ haplotypes from a perfect phylogeny without replacement n times. This model of data generation is not the same as the model studied in this paper, but their observations are clearly related. Let k be the number of distinct haplotypes. There seems to be a phase transition: when $n \ll k \log k$, there are many PPH solutions; when $n \gg k \log k$, there is typically only one. Cleary and St. John[13] then studied the structure of random pairing graphs; they showed that if there are $o(k \log k)$ population members, with high probability there are multiple PPH solutions. They also show experimentally, but do not prove, that above this bound, there is usually a unique PPH solution. Note that the number of sites and the number of distinct haplotypes are directly correlated in the coalescent model. This observation explains the dependence on length of the sequences observed in their experiments.

3. LINEAR ALGEBRAIC STRUCTURE AND A FIRST BOUND

Our work focuses on the rank of the genotype matrix, which Kalpakis and Namjoshi[8] have previously noted is a lower bound on the size of the solution to the HIPP instance.

Lemma 3.1. *The number of haplotypes in the solution to the HIPP instance G is at least $k^* = \operatorname{rank}(G)$. If there exist a $k^* \times m$ haplotype matrix H and an $n \times k^*$ pairing matrix Π such that $G = \Pi \cdot H$, then H forms an optimal set of haplotypes for G.*

Proof. Since H is a valid set of haplotypes for G only if there exists a pairing matrix such that $G = \Pi \cdot H$, H must have rank at least $\operatorname{rank}(G)$, and must have at least that many rows. If $G = \Pi \cdot H$ and H has exactly k^* rows, then it matches the lower bound and is optimal. $\square$

Corollary 3.1. *If Π is an $n \times k^*$ pairing matrix of rank k^*, H is a $k^* \times m$ haplotype matrix of rank k^*, and $G = \Pi \cdot H$, then $k^* = \operatorname{rank}(G)$ and (Π, H) is an optimal HIPP solution for G.*

Proof. This follows since if both Π and H are full rank, so is their product. $\square$

3.1. The rank of the pairing and haplotype matrices

We now consider when we can expect G to be created from full-rank matrices H and Π.

3.1.1. *The rank of the pairing matrix*

Lemma 3.2. *If Π is a random pairing matrix for k haplotypes with more than $(\frac{1+\epsilon}{2})k \log k$ pairings, for some constant $\epsilon > 0$, then $\operatorname{rank}(\Pi) = k$ almost surely as $k \to \infty$.*

Proof. Π is the node-edge incidence matrix of the pairing multigraph. A standard graph theory result shows that Π is full rank if all connected components of the graph are non-bipartite[14]. For example, if the graph is connected and contains a triangle, then $\text{rank}(\Pi) = k$.

If the graph has $\frac{1+\epsilon}{2} k \log k$ pairings, then almost surely as $k \to \infty$ it has at least $\ell = \frac{1+\epsilon'}{2} k \log k$ distinct non-self pairings, for some constant $\epsilon' > 0$. As such, it contains a subgraph G' from the random graph model $G(k, \ell)$, with k nodes and ℓ edges, and each possible edge equally likely. The classic Erdős-Rényi Theorem[15] shows that for such ℓ, G' is connected almost surely, and a standard textbook exercise (see, for example, Bollobas's textbook[16]) gives that such a graph contains a triangle almost surely, as $k \to \infty$. As such, Π is rank k almost surely as $k \to \infty$. □

3.1.2. *Pairing in non-uniform haplotype pools*

We may also be sampling from a non-uniform population, where some haplotypes are more common than others. If there is a pool of p different haplotypes being randomly paired to form genotypes, but only k distinct haplotypes, we may require substantially more genotypes in order to be confident that we have paired all of the haplotype kinds.

In this representation, we can assume that H, the haplotype matrix used to create G is a $p \times m$ matrix, and Π, the pairing matrix, is $n \times p$. If the matrix Π is rank p and the matrix H is rank k, then their product $G = \Pi \cdot H$ is also of rank k.

The Erdős-Rényi Theorem shows that as long as the number of distinct non-self pairings is at least $\frac{1+\epsilon}{2} p \log p$ (which is true, again, almost surely if the number of genotypes is $\frac{1+\epsilon'}{2} p \log p$, for some $\epsilon' > 0$), the pairing graph on the p nodes represented by the pool of p haplotypes is connected, and again, it will also contain a triangle as a subgraph as before. Hence, Π will be full rank (of rank p).

3.2. The haplotype matrix

Now we consider the haplotype matrix. It is of rank k when it comes from a perfect phylogeny.

Lemma 3.3. *If H is a $p \times n$ haplotype matrix with k distinct haplotypes and can be realized by a perfect phylogeny, with one column s_0 of all ones, then H is rank k.*

Proof. Each distinct haplotype h_i corresponds to a different leaf in the phylogenetic tree, and has value 1 at positions corresponding to mutations on the path from leaf to root. Consider two neighbouring leaves i and j in the subtree induced by haplotypes of H. Their sequences are distinct, so at least one of i or j has a mutation on the path from their common ancestor to it; suppose it is i. Then h_i has a one found in no other haplotype, and is thus linearly independent of all other haplotypes. Remove h_i and repeat this process for all k haplotypes. The column s_0 prevents a row of all zeros being the last row left, ensuring the matrix is of rank k, not $k-1$. Elementary column operations allow us to extend to the case where the root has allele 1 at other sites than s_0. □

3.3. Putting it together: a first bound

If the data are generated by a perfect phylogeny, and there are enough population members, then the rank of the genotype matrix *is* the optimal number of haplotypes.

Theorem 3.1. *Let G be a genotype matrix produced by random pairing of a pool of p haplotypes (not necessarily unique) that are generated by a perfect phylogeny (with a column of all ones). If G has at least $\frac{1+\epsilon}{2} p \log p$ pairings, then the size of the optimal solution to the HIPP instance G is* $\text{rank}(G)$*, almost surely as $p \to \infty$.*

Proof. The pairing matrix is almost surely of full rank, p, by Lemma 3.2, and the rank of the haplotype matrix equals its number of distinct haplotypes, by Lemma 3.3. Hence, Corollary 3.1 shows that the initial set of haplotypes found in H is an optimal set for this HIPP instance. □

Thus, for many instances, the general NP-hardness of HIPP is partly ameliorated: we can easily compute the size of the optimum, if not the actual haplotypes. In the next section, we see that for PPH instances with a few times more pairings can be exactly solved.

4. WHEN CAN WE FIND THE ACTUAL HAPLOTYPES?

For instances of the problem with a constant factor more genotypes than the bound of Theorem 3.1, and with a "common" mutation, we can almost surely identify the optimal haplotypes for PPH instances. We do this by connecting to a variant of the HIPP problem, where we restrict the haplotype matrix to be a PPH matrix. Our main result is the following constructive theorem.

Theorem 4.1. *Let G be derived from randomly pairing p haplotypes compatible with a perfect phylogeny, represented by the haplotype matrix H, with k distinct haplotypes. Suppose there exists a column of H (without loss of generality, s_1) with at least αp of both zeros and ones, for some $\alpha > 0$. If G arises from at least* $\max(\frac{1+\epsilon}{2}p\log p, \frac{2+\epsilon}{\alpha}p\log k)$ *pairings of members of H, then we can solve HIPP in polynomial time almost surely as $k \to \infty$ (and consequently as $p \to \infty$).*

For example, if $p = k$ and there is a site with minor allele frequency at least 25%, Theorem 4.1 says that if the number of rows of G is at least $(8+\epsilon)k\log k$, the optimal set of haplotypes can be found almost surely in polynomial time as $k \to \infty$.

We prove Theorem 4.1 through several steps. Our first lemma notes that we can restrict ourselves to PPH matrices H. The Min-PPH problem, studied by Bafna *et al.*[17], is the HIPP problem subject to this restriction on H. (Bafna *et al.* have shown that Min-PPH is NP-hard.)

Lemma 4.1. *If the conditions of Theorem 4.1 hold, then almost surely, the set of unique haplotypes in H is an optimal HIPP solution, and also an optimal solution to the Min-PPH instance G.*

Proof. The number of distinct population members is greater than $(\frac{1+\epsilon}{2})p\log p$, so Theorem 3.1 applies, and $\text{rank}(G) = k$ almost surely and the unique haplotypes of H form an optimal HIPP solution for G. Since they satisfy a perfect phylogeny, they are also a smallest PPH solution. □

Lemma 4.1 allows us to restrict our search to PPH solutions. Lemma 4.2 shows there exists only one, with high probability; Corollary 4.1 shows it can be found in polynomial time. This will complete the proof of Theorem 4.1.

Lemma 4.2. *Given a genotype matrix G satisfying the conditions of Theorem 4.1, almost surely there exists only one set of haplotypes that satisfies a perfect phylogeny and can generate G.*

Proof. We prove the lemma via a property of the DPPH algorithm of Bafna *et al.*[18]. This algorithm constructs a graph with one vertex for each column of G. The main result we use is that the number of PPH solutions for G is 2^{c-1}, where c is the number of connected components in a specific subgraph[18] of this graph. We show that if G satisfies the conditions of Theorem 4.1, then with high probability, $c = 1$, and there is only one PPH solution.

Graph $D(G)$ has one vertex for each distinct column in genotype matrix G and an edge between two vertices s and s' if there exists a row g of G with value 1 in columns s and s', and the resolution of g at sites s and s' is restricted by the perfect phylogeny condition. Recall from Section 2 that in our notation, 1 represents a heterozygous site. More precisely, we connect s and s' if we can find three genotypes g_1, g_2 and g_3 such that the 3×2 submatrix of G induced by these genotypes and sites has one of the forms in Figure 1. In each of these forms, one possible resolution of sites s and s' violates the perfect phylogeny condition, so the possible space of PPH solutions is restricted. The total number of PPH solutions for G equals 2^{c-1}, where c is the number of connected components of $D(G)$[18].

$$a)\begin{array}{cc}1 & 1\\ 1 & x\\ y & 1\end{array}\qquad b)\begin{array}{cc}1 & 1\\ 0 & 0\\ 2 & 2\end{array}\qquad c)\begin{array}{cc}1 & 1\\ 2 & 0\\ 0 & 2\end{array}$$

Fig. 1. Patterns of 3×2 submatrices which cause an edge to be added between the vertices representing the sites in $D(G)$. The values x and y are each either 0 or 2.

To show $D(G)$ is almost surely connected, we show that almost surely, there is an edge between each node and the node for s_1, the site with the common mutation. For any site, let e_i be the tree edge containing the mutation at site s_i, let A_i be the set of haplotypes below e_i and B_i be all other haplotypes. When considering the random pairings to make genotypes, we use $\langle X, Y\rangle$ to denote a pairing of a haplotype from class X with a haplotype from class Y.

Consider a site s. We consider a few cases on s, depending on whether e_s is below e_1 or not, and on the size of A_s. First, suppose e_s is not below e_1. If A_s has fewer than $\frac{\alpha p}{2}$ haplotypes, we will have an edge between s and s_1 in $D(G)$ if the events $\langle A_s, A_1\rangle$, $\langle A_s, B_1 \setminus A_s\rangle$ and $\langle B_1 \setminus A_s, A_1\rangle$ occur, since these events produce a submatrix of type (a) in Figure 1. In a random pairing, each event has probability at least $\frac{\alpha}{2p}$. If $|A_s| > \frac{\alpha p}{2}$, the events $\langle A_s, A_1\rangle$, $\langle A_s, A_s\rangle$ and $\langle A_1, A_1\rangle$ produce a submatrix of type (c) in Figure 1. Again, all events have probability at least $\frac{\alpha}{2p}$.

Suppose instead e_s is below e_1. If $|A_s| < \frac{\alpha p}{2}$, the events $\langle A_s, B_1\rangle$, $\langle A_s, A_1 \setminus A_s\rangle$ and $\langle B_1, A_1 \setminus A_s\rangle$ give a submatrix of form (a), while if $|A_s| \geq \frac{\alpha p}{2}$, the events $\langle A_s, B_1\rangle$, $\langle A_s, A_s\rangle$ and $\langle B_1, B_1\rangle$ give a submatrix of form (b). The needed events always have probability at least $\frac{\alpha}{2p}$.

Therefore, for each column s, three events each with probability at least $\frac{\alpha}{2p}$, will connect s and s_1. This totals less than $6k$ events, since there are at most $2k-2$ distinct columns in a perfect phylogeny with k distinct leaves. By the coupon-collector lemma[19], after $(\frac{2+\epsilon}{\alpha})p \log k$ random pairings, the probability that a needed event has not yet occurred is less than $6(k^{-\epsilon/2})$.

Thus, if the conditions of Theorem 4.1 are satisfied, then almost surely, $D(G)$ is connected and the DPPH results of Bafna *et al.*[18] show that there exists a unique PPH solution for G. □

Corollary 4.1. *A Min-PPH instance G satisfying the conditions of Theorem 4.1 can be solved in polynomial time with high probability.*

Proof. The algorithm DPPH of Bafna *et al.*[18] gives a representation of all PPH solutions for a given genotype matrix G in polynomial time, and allows their enumeration in time polynomial in the input matrix size and proportional to the number of PPH solutions. Lemma 4.2 shows that there is a unique PPH solution almost surely, and it can be recovered in polynomial time. □

4.1. A bound on finding a common mutation in a coalescent model

Our theorems show that almost surely, if there exists a mutation with minor allele frequency at least α in our data, we will likely be able to solve HIPP if the number of genotypes is above a relatively small bound. How often does such a common mutation occur? One would likely not study a population if all mutations were rare, but we can also give a partial answer to this question probabilistically, in a simple version of the coalescent model from population genetics.

Our results show that to guarantee that with probability $1-\epsilon$ there is a site chosen with minor allele frequency at least α, one needs to set the parameter θ in the coalescent process to a constant depending only on α and ϵ; hence, the number of polymorphic sites needs only increase as the logarithm of the population sample's size. Our bounds are coarse, but again prove that one can have provably high success in solving synthetic HIPP instances.

The question of how many mutations with minor allele frequency α can be expected to exist has been studied by theoretical population geneticists; see, for example, Fu[20]. However, the work of these authors has mostly concerned the expected number of mutations with minor allele frequency α; we need to determine how large θ must be in order to be confident that a mutation of the type we desire exists *almost surely.*

4.1.1. *An introduction to the coalescent model*

Infinite-site constant-population coalescent models are a standard population genetics model to produce haplotypes. We describe them briefly, focusing on details we need; for full detail, see Hein *et al.*[21]. In particular, we focus on the event order, not the time between events; see Hudson[22] for justification of this approach. The coalescent approach is used, for example, in the program `ms`[23], which has been used by several groups to generate HIPP problem test instances, by randomly pairing the resultant haplotypes[1, 5, 7].

The coalescent model describes the descent of a population under neutral evolution. We use it to generate rooted trees with p leaves, where each leaf represents a haplotype. We will describe the model going backward in time: we begin with the p leaves, and coalesce them to their common ancestor. Two kinds of events can occur as we move backwards in time: mutations and coalescences; the parameter θ governs which of these is more likely to happen. (In population genetics, $\theta = 2N\mu$ depends on both μ, the mutation rate, and N, the effective population size.) If k lineages are active at a point in time, a coalescence is the next event with probability $\frac{k-1}{k-1+\theta}$, and a mutation with probability $\frac{\theta}{k-1+\theta}$. When a mutation is indicated, one of the active lineages is uniformly chosen, a mutation at a new polymorphic site is assigned to it. When a coalescence is indicated, two lineages are uniformly chosen and joined into a common ancestor. The process continues until only one lineage remains, which is the common ancestor for all sites. All random choices are independent. We establish haplotypes for the sequences, as in Section 2.3.

We can sample from the same distribution of tree topologies by thinking about the coalescent process by moving forward in time, not backward. The standard way to do this is as a branching process, where we start with one lineage, and then whenever a divergence event is indicated, a lineage is chosen from all of the i lineages, and it bifurcates to produce $i+1$ lineeages; this process is continued until there are n lineages present. (Mutations occur in this process as in the backward conception of the coalescent, but we can ignore this here. We will use the forward branching process model only to estimate the number of lineages at a time when a mutation occurring on one lineage in particular would be sufficient to create a mutation of our desired type; we then switch to the backward coalescent version of the process, conditioned on having a lineage of our desired type.)

For our purposes, however, it is actually easier to use a non-standard forward-in-time way to sample from this distribution. A fact about the branching process is that if we pick one of the two lineages that result from the initial bifurcation, its number of eventual descendants in the n-member population is uniformly distributed over $\{1, \ldots n-1\}$ (see, *e.g.* Ref 22.) Moreover, the structure of the two trees that result from this bifurcation, one with i eventual descendants and the other with $n-i$, are themselves chosen independently from the coalescent distribution with those number of nodes.

As such, we can generate trees from the coalescent distribution in a somewhat different-appearing formulation that is still equivalent, by annotating each lineage with the number of eventual descendants that it will have; the initial lineage is annotated to have n eventual descendants. We still pick our branching lineage uniformly at random from the active lineages, but a lineage is only active if it has more than one eventual descendant; those with only one descendant will never bifurcate again. When we choose a lineage with i eventual descendants to bifurcate, we sample the number of descendants that one of the new lineages will have uniformly from $\{1, \ldots, i-1\}$, with the other new lineage resulting form the bifurcation having i minus that many descendants. After $n-1$ such bifurcations, we will have chosen the topology of our coalescent tree. Since we are successively conditioning according to the probabilities of the traditional forward coalescent process, the tree we choose by this procedure is chosen from the same probabilistic distribution as for the classic model.

4.1.2. *Common mutations*

We now connect the coalescent model to our HIPP theorems. If we use the model to generate p haplotypes, we can apply Theorem 4.1, if we have a polymorphic site with minor allele found in at least αp haplotypes.

Theorem 4.2. *Suppose that we produce p haplotypes by the coalescent model of Section 4.1.1. For any $\epsilon > 0$, if we choose the parameter θ of the coalescent model to be at least $\frac{1-\epsilon}{\epsilon}(\epsilon^{1/(2\alpha-1)} - 1)$, then with probability at least $1 - 2\epsilon$, we will have a site with minor allele frequency at least α. Also, if θ is $\omega(1)$ as a function of p, such a site is chosen almost surely as $p \to \infty$.*

We prove Theorem 4.2 by focusing on finding one edge with between αp and $(1-\alpha)p$ descendants (a "good" edge) with a mutation. Our bounds are likely coarse as a consequence. First, we show that there is likely a good edge reasonably high in the tree.

Lemma 4.3. *Consider a coalescent tree with p leaves, and assume $0 < \alpha < 1/3$. The probability that the top of a good edge exists in the tree at or before the ℓth bifurcation from the top is at least*

$1-\ell^{2\alpha-1}$. *Given $\epsilon > 0$, with probability at least $1-\epsilon$, there is such an edge with start at or above the $\ell^*_{\alpha,\epsilon}$-th bifurcation, for $\ell^*_{\alpha,\epsilon} = \epsilon^{1/(2\alpha-1)}$.*

Proof. At each step, there exists a lineage with the most descendants. If it has fewer than $(1-\alpha)p$ descendants, we have already seen a good edge. To see this, consider the first time this happens: we divided a value greater than $(1-\alpha)p$ into two parts, both smaller than $(1-\alpha)p$. Since $\alpha < 1/3$, one is at least αp.

Thus, we can concern ourselves with bifurcations on the lineage with the most descendants. At step i in the coalescent process, going forward, this lineage has probability at least $1/i$ of being chosen to bifurcate; it may be more if there are lineages with only one descendant haplotype. If it bifurcates, the probability of a good edge being produced is at least $1-2\alpha$. Since all bifurcations are independent, we can upper bound the probability of no good edges occurring by level ℓ by $\prod_{i=1}^{\ell}(1-\frac{1-2\alpha}{i}) < \prod_{i=1}^{\ell} e^{\frac{2\alpha-1}{i}} < e^{(2\alpha-1)\log \ell} = \ell^{2\alpha-1}$. The bound as a function of the probability $1-\epsilon$ of a good edge at or above level $\ell^*_{\alpha,\epsilon}$ is easily shown by arithmetic. □

Now, we can finish the proof of Theorem 4.2.

Proof. By Lemma 4.3, with probability at least $1-\epsilon$, there is a period in the coalescent history of the sequences during which there are fewer than $\ell_{\alpha,\epsilon}$ lineages, and where a mutation on one lineage would be a good mutation. Sticking to such instances, and now working backward in time, the next event on that lineage is a mutation with probability at least $\frac{\theta}{\ell_{\alpha,\epsilon}-1+\theta}$; if there are more coalescences of other parts of the tree before an event on our good lineage, it only increases the probability of mutation preceding coalescence on the lineage of interest. Setting this probability equal to $1-\epsilon$ and solving for θ gives that if $\theta \geq \frac{(1-\epsilon)}{\epsilon}(\ell_{\alpha,\epsilon}-1)$, then the probability of a mutation of minor allele frequency at least α is at least $1-2\epsilon$. □

Our bounds, while perhaps odd, are constant as a function of p. They indicate how far down in the tree one must look in order to be guaranteed a high probability of having found a good edge, and this depends solely on α. Since the expected number of mutations in the tree is approximately Poisson distributed with mean $\theta \log p$ (see Ref 21), we note that if the number of mutations accumulated is $\omega(\log p)$, then a common mutation exists almost surely (for any α) as p grows.

4.2. Applicability to small populations

The previous theorems are asymptotic results depending on the value of p. However, a small experiment shows that they apply for small p as well. For a variety of values of p, we used Hudson's program `ms`[23] to generate a PPH matrix with varying values of the mutation parameter θ, and paired the haplotypes randomly to generate n distinct genotypes. Shown in Table 1 are the number of genotypes needed so that in 200 experiments, the generating set of haplotypes (after removing duplicates) was always optimal for HIPP and was the unique Min-PPH solution. (We verified the number of PPH solutions using Ding, Filkov and Gusfield's LPPH[24].) Even for moderate values of p and θ, $1.1p \log p$ genotypes satisfied these conditions.

Table 1. The smallest number of genotypes n for which all 200 trials passed the rank and PPH tests.

	Number of haplotypes p							
θ	10	15	30	50	75	100	150	200
5	25	45	60	75	130	115	150	185
10	25	40	85	130	155	195	260	310
20	35	45	90	160	220	260	440	515
40	35	55	110	200	295	395	540	675

5. THE ALGEBRAIC RANK OF NON-PPH INSTANCES

For PPH instances, rank(H) exactly equals its number of unique haplotypes. If Π is full rank, then the unique rows of H form an optimal solution to the HIPP instance $G = \Pi \cdot H$ with exactly rank(G) haplotypes. For instances generated by models that include recombination, the situation is more complicated: H may not be full rank, and the rank of G may not equal the size of its optimal solution. We now study ranks of haplotype matrices in such models, assuming always that Π is of full rank. We use the rank of H to prove a lower bound on the number of recombinations in a phylogenetic network that explains a set of haplotypes, which is provably at least as strong as the commonly used "haplotype lower bound"[9]. For *galled trees*, a class of recombination networks, we give a full characterization of the rank. One interesting feature of our findings is that estimating the number of recombinations and performing haplotype inference by pure parsimony seem opposed to each other. We discuss this more in Section 5.4.

5.1. Phylogenetic networks with recombination

We first give a combinatorial description of this domain. A phylogenetic network is a rooted directed acyclic graph with edges pointing away ("down") from the root. The leaves (indegree 1 and outdegree 0) correspond to current haplotypes. Coalescent nodes (indegree 1 and outdegree 2) correspond to the most recent common ancestor of their descendant haplotypes. Recombination nodes (indegree 2 and outdegree 1) correspond to when two incoming lineages recombine to form a single lineage, a prefix of one lineage followed by a suffix of the second lineage. The node is labelled to indicate which parental lineage takes each role and the discrete *recombination breakpoint* where the recombination occurs.

Mutations are assigned to edges of the network. Each mutation has a chromosomal position, which mutates once, from allele 0 to allele 1, in the network. We assume that the haplotype at any position in the network identifies the allele found at that network position for every site with a mutation in the network. Without loss of generality, we assume that at the top of the network, the haplotype is all zeros, except for a one in a special site s_0 that never mutates. At a coalescent node or a leaf, the haplotype is the haplotype at the parent of the node, with zeros changed to ones at positions corresponding to any mutations on the edge separating them. At recombination nodes, labelled with position k, the first k sites of the haplotype come from the parent corresponding to the prefix edge into the node and the remainder comes from the other parent.

An important element of a phylogenetic network are the recombination cycles between recombination nodes and coalescent nodes. For any network, we can give a partial order over recombination cycles, when the recombination node of one is a descendant of another, and then identify an order from the bottom of the network to the top.

Gusfield *et al.*[25] defined a simple kind of recombination network, galled trees, in which every edge is found in at most one recombination cycle. We will focus our attention on the rank of data coming from a galled tree, but first give a general rank bound.

5.2. The rank of data from phylogenetic networks

We can easily relate the data rank to the number of recombinations in the network.

Theorem 5.1. *Let H be a haplotype matrix with k unique rows derived from a phylogenetic network with r recombinations. Then* $\mathrm{rank}(H) \geq k - r$. *Stated another way,* $r \geq k - \mathrm{rank}(H)$.

Proof. We prove this by induction on the number of recombinations in the network. If the network has no recombinations, it is a tree and has a column of all ones, so Lemma 3.3 applies. If not, consider a lowest recombination node. Below it is a tree; suppose its leaves have p unique haplotypes. If there is a mutation found only in all p haplotypes, then the lemma applies and removing the p haplotypes drops the rank by p. If no such mutation exists, removing the p haplotypes drops the rank by at least $p-1$, but not necessarily p. In either case, we remove that recombination node and its descendants, and have a network with one fewer recombination. □

This rank bound may be surprisingly useful; it is similar in spirit to the "haplotype lower bound"[9] on the number of recombinations required to explain a haplotype matrix, which equals $k - c + 1$, where c is the number of unique columns in the matrix H. The haplotype bound is often negative, because there may be many different columns, but $k - \mathrm{rank}(H)$ is always non-negative, and thus may be stronger. We also note that this bound applies for unknown ancestral sequence, as it can be adjusted in the standard way to apply to the case of a known ancestral sequence. Of course, $\mathrm{rank}(H)$ is slower to compute than the number of distinct columns of H, but use of the bound may still be interesting to explore.

5.3. The algebraic rank of galled trees

The rank of the haplotype matrix can decrease from full rank by at most the number of recombinations in the network that gives rise to the haplotypes. In the case of galled trees, we can identify for each recombination whether it actually does reduce the rank or not.

We will inspect recombination cycles from the bottom of the network up, and identify them as rank-decreasing, rank-maintaining, or rank-confounding. For rank-confounding cycles, all haplotypes in the cycle are independent, but there may be a dependency between them and the other haplotypes in the tree, so we add a new haplotype to determine this.

There are three types of node in the recombination cycle. The coalescent node and recombination node that together define the recombination cycle will simply be referred to as the *coalescent node* and *recombination node*, respectively. The other nodes in the cycle (although marking coalescence events) will be referred to as *cycle nodes.* In a recombination cycle, a node is *included* if it represents a haplotype in H, and no higher node on the cycle also represents that haplotype. The sides of the cycle are the two directed paths from the coalescent node to the recombination node. Consecutive included nodes are nodes on either side of the cycle that have only unincluded nodes between them. Let h_c be the haplotype at the coalescent node and h_r be the haplotype at the recombination node.

Theorem 5.2. *A recombination cycle is rank-maintaining if:*

- *the recombination node is not included, or*
- *the recombination node is included and has a mutation found in no other included node in the cycle, or*
- *between the coalescent node and the first included cycle node on either side of the cycle, or between any two consecutive included nodes on the cycle are found two mutations on either side of the recombination breakpoint of the cycle (a "rank-maintaining pair").*

A recombination cycle is rank-decreasing if it is not rank-maintaining, and the coalescent node of the cycle is included.

A recombination cycle is rank-confounding if it is neither rank-maintaining nor rankdecreasing. The rank of H will be the same as that of H with h_r removed and h_c added.

Proof. We need to detect whether the haplotype h_r at the recombination node is independent of all other haplotypes. We begin with the rank-maintaining cases. The case where h_r is not included is trivial. If there is a mutation j unique to h_r among cycle nodes, that column is independent of all other columns, so the recombination node is independent; other haplotypes may possess mutation j, but they also possess other mutations not found in h_r. If there is a rank-maintaining pair, then every included haplotype that possesses one mutation from the pair also possesses the other, *except* h_r, so h_r is independent.

If there is no rank-maintaining pair, all mutations on the cycle can be individually isolated by subtracting haplotypes at consecutive included nodes. Elementary row operations can thus transform h_r into h_c. If h_c is already in H, then h_r is not independent of the other haplotypes in H; if it is not, it may or may not be independent. Wwe must add h_c to our set of haplotypes find out. □

Going through each cycle obeying the partial order, we can identify its effect on the rank, and thus compute the overall rank of H.

5.4. Consequences of the rank bounds for phylogenetic networks

In the standard variation of the coalescent process that includes recombination, the relative rates of recombination and mutation are given by two parameters ρ and θ. When ρ is large relative to θ, recombination is common, whereas when θ is large, recombination is rare.

The bounds from Theorems 5.1 and 5.2 can be read to say that if mutation is common relative to recombination, the rank of H (and consequently $G = \Pi \cdot H$, if Π is full rank) is likely to be close to its number of unique haplotypes; many mutations will make it likely that the haplotype at a recombination node is linearly independent of the other haplotypes. When rank is high, the most parsimonious set of haplotypes to produce the genotype matrix G is likely to have close to the same number of distinct haplotypes as does H.

By contrast, when recombination is more common than mutation, we may start to accumulate many rank-decreasing cycles. This may mean that for the genotype matrix G, the rank bound on the HIPP solution may be far from optimal. But, interestingly, for the haplotype matrix H, the lower bound on the minimum number of recombinations to explain H will be increasingly accurate, since this bound goes up as the rank goes down.

This suggests a tension between the HIPP problem and estimating the number of recombinations: for instances with few recombinations, we get little information about the minimum number of recombinations, but may obtain a close match on the minimum number of haplotypes. For instances with lots of recombinations, we get no information about the minimum number of haplotypes, but may get some information about the number of recombinations. Unfortunately, we cannot identify which of these bounds we have, just from the rank of G.

6. CONCLUSION

We have presented several results about algebraic rank and HIPP instances, studying HIPP instances generated by randomly pairing haplotypes generated from two important models from population genetics. For data generated by a perfect phylogeny on p haplotypes, when the number of distinct population members is more than $(\frac{1+\epsilon}{2})p \log p$, the size of the optimal solution equals the rank of the genotype matrix. Moreover, with only a few times more genotypes and a common mutation, we can recover the haplotypes in polynomial time.

We studied more closely data generated by the coalescent model often studied in population genetics; this is relevant, for example, for data generated with Hudson's popular `ms` package[23]. We showed that the constant value for the mutation parameter θ, to guarantee a common mutation with probability $1 - \epsilon$.

Finally, we examined the effect of adding recombination to generative model. Here we derive two interesting results. First, we provide an interesting variant of the "haplotype lower bound". This shows that rank is still a close bound when the model allows recombination. Second, we completely classify the algebraic rank of haplotype matrices derived from galled trees, the simplest type of phylogenetic network with recombination. The algebraic structure of HIPP instances will likely have other fruitful consequences as well.

Acknowledgements

This research has been supported by the Natural Sciences and Engineering Research Council of Canada, through a Discovery Grant to D.B. and Postgraduate and Canada Graduate Scholarships to I.H. We would like to thank Katherine St. John for sending us a copy of her paper with Sean Cleary[13], and Dan Gusfield, Yun Song and Brian Golding for helpful conversations.

References

1. D. Gusfield. Haplotype inference by pure parsimony. In *Proceedings of CPM 2003*, pages 144–155, 2003.
2. The International HapMap Consortium. A haplotype map of the human genome. *Nature*, 437(7063):1299–1300, 2005.
3. K. Lindblad-Toh et al. Genome sequence, comparative analysis and haplotype structure of the domestic dog. *Nature*, 438(7069):803–809, 2005.
4. G. Lancia, C. M. Pinotti, and R. Rizzi. Haplotyping populations by pure parsimony: Complexity of exact and approximation algorithms. *INFORMS Journal on Computing*, 16:348–359, 2004.
5. D. G. Brown and I. M. Harrower. A new integer programming formulation for the pure parsimony problem in haplotype analysis. In *Proceedings of WABI 2004*, pages 254–265, 2004.
6. L. Wang and Y. Xu. Haplotype inference by maximum parsimony. *Bioinformatics*, 19(14):1773–1780, 2003.
7. D.G. Brown and I.M. Harrower. Integer programming approaches to haplotype inference by pure parsimony. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(2):141–154, 2006.
8. K. Kalpakis and P. Namjoshi. Haplotype phasing using semidefinite programming. In *Proceedings of BIBE 2005*, pages 145–152, 2005.
9. S.R. Myers and R.C. Griffiths. Bounds on the minimum number of recombination events in a sample history. *Genetics*, 163:375–394, 2003.
10. J. He and A. Zelikovsky. Linear reduction for haplotype inference. In *Proceedings of WABI 2004*, pages 242–253, 2004.
11. B. V. Halldórsson, V. Bafna, N. Edwards, R. Lippert, S. Yooseph, and S. Istrail. A survey of computational methods for determining haplotypes. In *Com-*

putational Methods for SNPs and Haplotype Inference: DIMACS/RECOMB Satellite Workshop, volume 2983 of *LNCS*, pages 26–47, 2004.

12. R. H. Chung and D. Gusfield. Empirical exploration of perfect phylogeny haplotyping and haplotypers. In *Proceedings of COCOON 2003*, pages 5–19, 2003.
13. S. Cleary and K. St. John. Analyses of haplotype inference algorithms. 2005. Manuscript under review.
14. C. Van Nuffelen. On the incidence matrix of a graph. *IEEE Transactions on Circuits and Systems*, 23(9):572–572, Sep 1976.
15. P Erdős and A Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.
16. B. Bollobás. *Random Graphs*. Cambridge Press, 2nd edition, 2001.
17. V. Bafna, D. Gusfield, S. Hannenhalli, and S. Yooseph. A note on efficient computation of haplotypes via perfect phylogeny. *Journal of Computational Biology*, 11:858–866, 2004.
18. V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: A direct approach. *Journal of Computational Biology*, 10:323–340, 2003.
19. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge Press, 1995.
20. Y. X. Fu. Statistical properties of segregating sites. *Theoretical Population Biology*, 48(2):172–177, 1995.
21. J. Hein, M. H. Schierup, and C. Wiuf. *Gene Genealogies, Variation and Evolution*. Oxford University Press, 2005.
22. R. R. Hudson. Gene genealogies and the coalescent process. *Oxford Surveys of Evolutionary Biology*, 7:1–44, 1990.
23. R. R. Hudson. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics*, 18(2):337–338, 2002.
24. Z. Ding, V. Filkov, and D. Gusfield. A linear-time algorithm for the perfect phylogeny haplotyping (pph) problem. In *Proceedings of RECOMB 2005*, pages 585–600, 2005.
25. D. Gusfield, S. Eddhu, and C. Langley. Optimal, efficient reconstruction of phylogenetic networks with constrained recombination. *Journal of Bioinformatics and Computational Biology*, 2(1):173–213, 2004.