EFFICIENT COMPUTATION OF MINIMUM RECOMBINATION WITH GENOTYPES (NOT HAPLOTYPES)

Yufeng Wu [*] and Dan Gusfield

Department of Computer Science
University of California, Davis
Davis, CA 95616, U.S.A.
Email: {wuyu,gusfield}@cs.ucdavis.edu

A current major focus in genomics is the large-scale collection of genotype data in populations in order to detect variations in the population. The variation data are sought in order to address fundamental and applied questions in genetics that concern the haplotypes in the population. Since almost all the collected data is in the form of genotypes, but the downstream genetics questions concern haplotypes, the standard approach to this issue has been to try to first infer haplotypes from the genotypes, and then answer the downstream questions using the inferred haplotypes. That two-stage approach has potential deficiencies, giving rise to the general question of how well one can answer the downstream questions using genotype data without first inferring haplotypes, and also giving rise to the goal of computing the *range* of downstream answers that would be obtained over the range of possible inferred haplotype solutions. This paper provides some tools for the *study* of those issues, and some partial answers. We present algorithms to solve downstream questions concerning the minimum amount of recombination needed to derive given *genotypic* data, without first fixing a choice of haplotypes. We apply these algorithms to the goal of finding recombination hotspots, obtaining as good results as a published method that first infers haplotypes; and to the case of estimating the minimum amount of recombination needed to derive the true haplotypes underlying the genotypic data, obtaining weaker results compared to first inferring haplotypes using the program PHASE. Hence our tools allow an initial study of the two-stage versus one-stage issue, in the context of specific downstream questions, but our experiments certainly do not fully resolve the issue.

1. INTRODUCTION

The field of genomics is now in a phase where large-scale data in populations is collected in order to study population-level variations [8]. Variations between individuals are used to provide insight into basic biological processes such as meiotic recombination, or to locate genes that are currently under natural selection [36], and to help locate the genes that influence genetic disease or economic traits (through a technique called "association mapping" or "LD mapping") [15]). Algorithms and computation play a central role in all of these efforts, and there is a growing literature on several key problems involved in both the acquisition and the downstream analysis of population variation data. In discussing acquisition and analysis problems, and in order to introduce the theme of this paper, we must first define some basic terms, concepts and issues.

In diploid organisms (such as humans) there are two (not completely identical) "copies" of each chromosome, and hence of each region of interest. A description of the data from a single copy is called a *haplotype*, while a description of the conflated (mixed) data on the two copies is called a *genotype*. Today, the underlying data that forms a haplotype is usually a vector of values of m *single nucleotide polymorphisms (SNP's)*. A SNP is a single nucleotide site where exactly two (of four) different nucleotides occur in a large percentage of the population. Genotype data is represented as an n by m 0-1-2 (ternary) matrix G. Each row is a genotype. A pair of binary vectors of length m (haplotypes) *generate* a row i of G if for every position c both entries in the haplotypes are 0 (or 1) if and only if $G(i,c)$ is 0 (or 1) respectively, and exactly one entry is 1 and one is 0 if and only if $G(i,c) = 2$. The international Haplotype Map Project [7, 8] is focused on determining, both molecularly and computationally, the common haplotypes in several diverse human populations.

The Key Issue The key technological fact is that it is very difficult and costly to collect large-scale haplotype data, but relatively easy and cheap

[*]Corresponding author.

to collect genotype data. But mutation, recombination, selection, evolution, all operate on haplotypes, not genotypes, and therefore the "downstream" biological questions that we want to answer using population variation data (for example, about recombination hotspots, linkage disequilibrium, natural selection, association mapping, phylogenetic networks, etc.) are all questions that are most naturally framed in terms of haplotypes. So, we have the *ability* to gather large-scale genotypes in populations, but we have the *need* to ask and answer questions about the underlying haplotypes in populations. To date, the resolution of this issue has overwhelmingly involved two independent stages: First, try to infer the "correct" haplotypes from the genotypes, either inferring a pair of haplotypes for each genotype in the sample, or inferring just the frequencies of the haplotypes in the sample; Second, do the downstream analysis using those inferred haplotypes.

There is a very large literature on haplotype inference (HI), and on an absolute scale, the underlying haplotypes can be inferred with remarkable fidelity [25], although problems remain and the field of haplotype inference is still very active (for example, even the developer of the most widely used program PHASE [35] has recently introduced a totally different approach in order to address much larger data than PHASE can handle [33]). So, the two-stage approach may in the end be the best way to address many of the downstream biological questions of interest, but in general there is some (potential) loss of information in any two-stage approach, and certainly this particular approach has both problems and missed opportunities. The main problem is that the haplotype inferences are likely to be incorrect to some extent and is not clear what effect those inaccuracies will have on the downstream analysis. The missed opportunities inherent in the two-stage approach is that by choosing just one set of haplotypes, we do not address questions about the *range* of possible answers to the downstream questions that the collected genotype data support. Range questions are of interest because they provide a kind of "sensitivity analysis" for any particular chosen answer (for example for an answer derived from the two-stage approach), and they address the general question of "how much does it really help to know the underlying haplotypes that gives rise to the genotypes" [6, 27, 28]? The answer to that question helps determine how much effort or money one would be willing to spend to determine the correct haplotypes (by molecular means or by gathering more genotype data). Indeed, we are seeing results in this general direction. For example, Halperin, et al. developed a method for tag SNP selection with genotype data [14].

The Main Theme of This Paper: Motivated by the above discussion, this paper concerns the solution to certain downstream biological questions using *genotypic* data, without first fixing a choice of haplotypes. In particular, we are concerned with estimating, and bracketing the range of the minimum amount of recombination needed to derive haplotypes that can pair to form the observed genotypes, and with problems of inferring an explicit evolutionary history of haplotypes that can pair to form the observed genotypes. As a byproduct, turning the two-stage process on its head, we can use some of these computations to solve the haplotype inference problem itself. We develop polynomial time algorithms for some problems, non-polynomial but practical algorithms for other problems, and show the results of applying these methods to simulated and real biological data. Our methods provide some tools to study the two-stage versus one-stage issue, in the context of specific problems involving recombination. However, we do not claim that our experimental results resolve the issue of which approach is best.

2. ADDITIONAL DEFINITIONS

Before discussing our results in detail, we need some additional definitions.

Given an input set (or matrix) of n genotype vectors G of length m, the *Haplotype Inference (HI) Problem* is to find a set (or matrix) H of n pairs of binary vectors (with values 0 and 1), one pair for each genotype vector, such that each genotype vector in G is generated by the associated pair of haplotypes in H. H is called an "HI solution for G". Genotype data is also called "unphased data", and the decision on whether to expand a 2 entry in G to $[\substack{0\\1}]$ or to $[\substack{1\\0}]$ in H, is called a "phasing" of that entry. The way that all the 2's in a column are expanded is called the phasing of the column.

The standard assumption in population genetics [15, 16] is that at most one mutation has occurred in any sampled site in the evolution of the haplotypes. This is called the "infinite sites model". In addition to mutation, haplotypes may evolve due to recombination between haplotype sequences. Meiotic recombination takes two equal length sequences and produces a third sequence of the same length consisting of some prefix of one of the sequences, followed by a suffix of the other sequence. Meiotic recombination is one of the principal evolutionary forces responsible for shaping genetic variation within species. Efforts to deduce patterns of historical recombination or to estimate the frequency or the location of recombination are central to modern-day genetics [26], and recombination is at the heart of the logic of association mapping, a technique that is widely hoped to help locate genes influencing genetic diseases and important traits [15].

For a given set of haplotypes, computing the *minimum* number of recombinations needed to explain their evolution (under the infinite sites model) is a standard question of interest, for both practical and fundamental reasons. For a matrix of haplotypes H, we define $Rmin(H)$ as the minimum number of recombination events needed in a derivation of H from some unknown (or sometimes known) ancestral haplotype, under the infinite sites model. The problem of computing $Rmin(H)$ exactly is NP-hard, but there is a growing literature on polynomial-time methods that work on problems of special structure; on practical heuristics that are exact on small data; and on efficient methods to compute close bounds on $Rmin(H)$.

The evolutionary history of a set of haplotypes H, which evolve by site mutations (assuming the infinite sites model) and recombination, is displayed on a directed acyclic graph called a "Phylogenetic Network" or an "Ancestral Recombination Graph (ARG)". For a formal definition of these graphs, see Gusfield, et al. [12] or Gusfield [11].

In most of the results in this paper the concept of *site incompatibility* is fundamental. Given a haplotype matrix H, two sites (columns) p and q in H are said to be *incompatible* if and only if there are four rows in H where columns p and q contain all four of the ordered pairs 0,1; 1,0; 1,1; and 0,0. The test for the existence of all four pairs is called the "four-gamete test" in the population genetics literature. The classic Perfect Phylogeny theorem is that there is a phylogenetic network without recombination (and hence a tree), that derives haplotypes H, if and only if there is no incompatible pair of site in H. An HI solution, H, for G is a called a "PPH solution" if no pair of sites in matrix H are incompatible. The problem of determining if there is a PPH solution can be solved in linear time [9, 32].

3. RECOMBINATION LOWER BOUNDS OVER GENOTYPES

Let L denote a particular recombination lower bound *method* that works on haplotype data, and let $L(H)$ be the lower bound given by L when applied to haplotype matrix H. That is, $L(H) \leq$Rmin(H). Given a *genotype* matrix G, we define $MinL(G)$ as the *minimum* value of $L(H)$ taken over *every* HI solution H for G, that is, over all haplotype matrices that generate G. Similarly, we define $MaxL(G)$ by changing "minimum" to "maximum" in the definition. The two quantities, $MinL(G)$ and $MaxL(G)$, precisely define the *range* of results that method L will produce, over all possible HI solutions for G. Note that $MinL(G) \leq L(H^*) \leq Rmin(H^*)$ where H^* is the true (but unknown) set of haplotypes that gives rise to G, but it is not true that $Rmin(H^*) \leq MaxL(G)$. Rather, $L(H^*) \leq MaxL(G)$.

The motivation for wanting to know $MaxL(G)$ may be a bit unintuitive. One situation is where we are interested in the amount of recombination that must have occurred in the generation of the true haplotypes underlying the observed genotypes G, and the available tool for studying recombination levels is the ability to compute $L(H)$ given haplotypes H. An obvious question in this situation is whether it is valuable to expend additional resources to better determine more information about the true H (in the laboratory or by collecting more data). The difference $MaxL(G) - MinL(G)$ indicates the most that can be learned about recombination (through the use of $L(H)$), even with additional efforts to learn the true H. In particular, if the difference is small, determination of the true H has little value in this context, and if the difference is large, $MaxL(G)$ bounds the most that can be learned, even if the true H is known.

In the next three sections we develop lower bound methods that work on a genotype matrix G. These methods will also be useful in Section 4 where we develop a method to build a minimum ARG for a set of genotypes.

3.1. The case of the Hudson-Kaplan (HK) lower bound

The first and best-known lower bound on $Rmin(H)$ is the HK bound [19]. When L is the HK method, we use $MinHK(G)$ and $MaxHK(G)$ in place of $MinmL(G)$ and $MaxL(G)$. Previously, Wiuf [38] showed that $MinHK(G)$ can be computed in polynomial time. In this section, we show that $MaxHK(G)$ can also be computed in polynomial time. We first have to define the *incompatibility graph* and to briefly describe the HK bound and method.

The "incompatibility graph" $IG(H)$ for H is a graph containing one node for each site in H, and an edge connecting two nodes p and q if and only if sites p and q are incompatible. We will refer to a node of $IG(H)$ and to the site of G it corresponds to, interchangeably. The HK lower bound on $Rmin(H)$ can be described and computed as follows: Arrange the nodes of $IG(H)$ on a line in the order that the corresponding sites appear in the underlying chromosome. Then compute the *maximum* number of non-overlapping edges in the embedded graph $IG(H)$. Two edges that only share a single node are still non-overlapping. The computed number is the HK bound, denoted $HK(H)$. It is easy to establish that $HK(H) \leq Rmin(H)$, and that $HK(H)$ can be computed in time that is linear in the number of edges of $IG(H)$.

3.1.1. *Efficient algorithm for MaxHK(G)*

Given a genotype matrix G, we define the *maximal incompatibility graph* for G, denoted $MIG(G)$, as follows: Each node in $MIG(G)$ corresponds to a site in G, and there is an edge between nodes p and q if there *exists* an HI solution H for G so that the pair p, q is incompatible. Note that the existence of an edge (p, q) is determined *independently* of all other pairs of sites; the HI solution that is used for one pair can be completely different from the HI solution used for another pair. Therefore, we only need to look at sites p and q to determine if edge (p, q) is in $MIG(G)$. Graph $MIG(G)$ is a supergraph of every incompatibility graph $IG(H)$ where H is an HI solution for G. For example, suppose sites p and q in G are

$$\begin{array}{c} 0\,0 \\ 0\,1 \\ 1\,0 \\ 2\,2 \end{array}$$

Then there is an edge (p, q) in $MIG(G)$ because we can phase the 2's in row four as $[\begin{smallmatrix} 0\,0 \\ 1\,1 \end{smallmatrix}]$ to make p, q incompatible. We now describe the algorithm.

Algorithm $MaxHK$

1. Construct $MIG(G)$ for input data G.
2. Arrange the nodes of $MIG(G)$ on a line, in the order that the sites appear in the underlying chromosome. Then find a maximum-size set, E_G, of non-overlapping edges in $MIG(G)$. We claim that $|E_G| = MaxHK(G)$.

Time analysis: The first step takes $O(nm^2)$ time. The second time takes $O(m^2)$ time. Thus, the algorithm runs in $O(nm^2)$ time.

Correctness: For every HI solution H for G, $|E_G| \geq HK(H)$ because $IG(H)$ is subgraph of $MIG(G)$. Therefore, $|E_G| \geq MaxHK(G)$. To show the converse, it is sufficient to show that $|E_G| \leq HK(H)$ for *some* HI solution H for G. This is not immediate because it is not necessarily true that $MIG(G) = IG(H)$ for some HI solution H for G. But, if we can find an HI solution H for G where all the edges of E_G are in $IG(H)$ (where they will be non-overlapping), then $|E_G| \leq HK(H)$. The edges in E_G induce a graph, and consider one of the connected components, C, of that graph. Because the edges in E_G are non-overlapping and C is a connected component, the edges in C form a simple connected path along the nodes in C ordered from left to right in the embedded $MIG(G)$. Let $s_1, s_2, \ldots, s_k$ denote the ordered nodes in C. To construct the desired H, we first phase sites s_1, s_2 to make pair s_1, s_2 incompatible (that is possible since edge (s_1, s_2) is in $MIG(G)$). Now we move to site s_3. We want to make pair s_2, s_3 incompatible but we have already chosen how s_2 will be phased with respect to s_1. The critical observation is that this prior decision does not constrain the ability to make pair s_2, s_3 incompatible, although one has to pay attention to how s_2 was phased. In choosing how to phase s_3 relative to s_2, the only rows in G where a phasing choice has any effect on whether pair s_2, s_3 will be incompatible, are the rows where both those sites have value 2 in the genotype matrix G. For one such row k of G, suppose we need to phase the 2's in s_2, s_3 to produce the pair 0,1 or the pair 1,0 or both, in order to make pair s_1, s_2m incompatible. (The case where we need 0,0 and/or 1,1 is similar and omitted.) If column s_2 (for row k) has been phased as $[\genfrac{}{}{0pt}{}{0}{1}]$ we phase s_3 (for row k) as $[\genfrac{}{}{0pt}{}{1}{0}]$. Otherwise, we phase s_3 as $[\genfrac{}{}{0pt}{}{0}{1}]$. In either case, we will produce the needed binary pairs in sites s_2, s_3 for row k. Similarly, we can follow the same approach to phase sites $s_4, \ldots, s_k$, making each consecutive pair of sites incompatible.

In this way, we can construct a haplotyping solution H for G where all the edges of E_G (and possibly more) appear in $IG(H)$, and hence $|E_G| \leq HK(H) \leq MaxHK(G)$. But since $|E_G| \geq MaxHK(G)$, $|E_G| = MaxHK(G)$, completing the proof of the correctness of Algorithm MaxHK.

3.2. The case of connected-component lower bound

A "non-trivial" connected component, C, of a graph is a connected component that contains at least one edge. A trivial connected component has only one node, and no edges. For a graph I, we use $cc(I)$ to denote the number of non-trivial connected components in graph I. It has previously been established [13, 1] that for a haplotype matrix H, $cc(IG(H)) \leq Rmin(H)$, and that this lower bound can be, but is not always, superior to the HK bound when applied to specific haplotype matrices. Therefore, for the same reasons we want to compute $MinHK(G)$ and $MaxHK(G)$, we define $MinCC(G)$ and $MaxCC(G)$ respectively as the minimum and maximum values of $cc(IG(H))$ over every HI solution H for G. In this section we show that $MinCC(G)$ can be computed in polynomial time by Algorithm MinCC, using an idea similar to one used for $MaxHK(G)$. The problem of efficiently computing $MaxCC(G)$ is currently open.

Algorithm $MinCC$

1. Given genotype matrix G, construct graph $MIG(G)$ and remove all trivial components.
2. For each remaining component C, let $G(C)$ be the matrix G restricted to the sites in C. For each such C, determine if there is a PPH solution for $G(C)$, and remove component C if there is a PPH solution for $G(C)$.
3. Let K_c be the number of remaining connected components. We claim that $K_c = MinCC(G)$.

Time analysis: Constructing $MIG(G)$ takes $O(nm^2)$ time. Finding all components takes $O(m)$ time. Checking all components for PPH solutions takes $O(nm)$ time. Thus, the entire algorithm takes $O(nm^2)$ time.

Correctness. We first argue that $cc(IG(H)) \geq K_c$ for every HI solution H for G. Let H be an arbitrary HI solution for G, and consider one of the K_c remaining connected components, C, found by the algorithm. Since $G(C)$ does not have a PPH solution, there must be at least one incompatible pair of sites in H, and so at least one edge in C must also be in $IG(H)$. Further, since $IG(H)$ is a subgraph of $MIG(G)$, every connected component of $IG(H)$ must be completely contained in a connected component of $MIG(G)$. Therefore, there must be at least one non-trivial connected component of $IG(H)$ contained in C, and so $cc(IG(H)) \geq K_c$.

To finish the proof of correctness, it suffices to find an HI solution H' for G where $cc(IG(H')) = K_c$. Note that we can phase the sites in each connected component of $MIG(G)$ separately, assured that no pair of sites in different components will be made incompatible. This is due to the maximality of connected components, and the definition of $MIG(G)$. To begin the construction of H', for a non-trivial component C of $MIG(G)$ where $G(C)$ has a PPH solution, we phase the sites in C to create a PPH solution. As a result, none of those sites will be incompatible with any other sites in G. Next we phase the sites of one of the K_c remaining components, C, so that in H', the nodes of C form a connected component of $IG(H')$. To do this, first find an arbitrary rooted, directed spanning tree T of C. Then phase the site at the root and one of its children in T so that those two sites are made incompatible. Any other site can be phased as soon as its unique parent site has been phased. As in the proof of correctness for Algorithm MaxHK, and because each node has a unique parent, each site can be phased to be made incompatible with its parent site, no matter how that parent site was phased. The result is that all the sites of C will be in a single connected component of $IG(H')$, so $K_c \geq cc(IG(H')$. But $cc(IG(H)) \geq K_c$ for every HI solution H for G, so $MinCC(G) = K_c$, and the correctness of Algorithm MinCC is proved.

Final comments on the polynomial-time methods

Above, we developed polynomial-time methods to compute $MaxHK(G)$ and $MinCC(G)$, given genotypes G. These are two specific cases of our interest in efficiently computing $MinL(G)$ and $MaxL(G)$ for different lower bounding methods L that work on haplotypes. Clearly, for the best application of such numerical values, we would like to compute $MinL(G)$ and $MaxL(G)$ for the lower bound methods L that obtain the highest lower bounds on $Rmin(H)$ when given haplotypes H. The HK and the CC lower bounds are not the best, but are of interest because they allow provably polynomial-time methods to compute $MinHK(G), MaxHK(G)$ and $MinCC(G)$. Those results contribute to the theoretical study of lower bound methods, and may help to obtain polynomial-time, or practical methods, for better lower bound methods. In the next section we discuss a practical method (on moderate size data) to compute better lower bounds given genotypes.

3.3. Parsimony-based lower bound

One of the most effective methods to compute lower bounds on $Rmin(H)$, for a haplotype matrix H, was developed in Myers, et al. [30], further studied in Bafna, et al. [2], and optimized in Song et al [34]. All of the methods in those papers produce lower bounds on $Rmin(H)$ that are much superior to $HK(H)$ and $CC(H)$, particularly when $n > m$. Therefore, given G, we would like to compute the minimum and/or maximum of these better bounds over all HI solutions for G. Unfortunately, we do not have a polynomial-time method for that problem, and we presently solve it only for very small data. However, we have developed a lower bounding method that works on genotype matrices of moderate size, using an idea *related* to the cited methods, and we have observed that when $n > m$, the lower bound obtained is often much superior to $MinHK(G)$ and $MinCC(G)$.

All the lower bound methods in the papers cited above work by first finding (local) lower bounds for (selected) intervals or subsets of sites in H, and then combining those local bounds to form a *composite* lower bound on $Rmin(H)$. The composition method was developed in Myers, et al. [30] and is the same for all of the methods. What differs between the methods is the way local bounds are computed. We do not have space to fully detail the methods, but all the local bounds are computed with some variation of the following idea [30]: Let $Hap(H)$ be the number of distinct rows of H, minus the number of distinct columns, minus 1. Then $Hap(H) \leq Rmin(H)$. $Hap(H)$ is called the *Haplotype lower bound.* When applied to the entire matrix H, $Hap(H)$ is often a very poor lower bound, but when used to compute many local lower bounds in small intervals, and these local bounds are combined with the composition method, the overall lower bound on $Rmin(H)$ is generally quite good.

Similar to the methods that work on haplotype data, given a genotype matrix G, we compute relaxed Haplotype lower bounds for many small intervals, and then use the composition method to create an overall number $Ghap(G)$ which is a lower bound on the minimum $Rmin(H)$ over every HI solution H for G. Of course, to be of value, it must be that $Ghap(G)$ is larger than $MinHK(G)$ and $MinCC(G)$ for a large range of data.

We now explain how we compute the local bounds in G that combine to create $Ghap(G)$. When restricted to sites in an interval, we have a submatrix G' of G. An HI solution H' for a genotype matrix G' is called a "pure parsimony" solution if it minimizes the number of distinct haplotypes used, over all HI solutions for G'. If the number of distinct haplotypes in a pure parsimony HI solution for G' is $p(G')$, and G' has m' sites, it is easy to show that $p(G') - m' - 1 \leq Rmin(H')$ for any HI solution H' for G'. We call this bound $Par(G')$. To compute $Ghap(G)$, we compute the local bound $Par(G')$ for each submatrix of G defined by an interval of sites of G, and then combine those local bounds using the composition method from Myers, et al [30]. It is easy to show that $Ghap(G) \leq Rmin(H)$ for every HI solution H for G.

The problem of computing a pure parsimony haplotyping solution is known to be NP-hard [17, 22], so computing $Par(G')$ is also NP-hard. But, a pure parsimony HI solution can be found relatively efficiently in practice on datasets of moderate size by using integer linear programming [10]. Other papers have shown how to solve the problem on larger datasets [4, 5]. Therefore, each local $Par(G')$ bound can be computed in practice when the size of G' is moderate, and so $Ghap(G)$ can be computed in practice for a wide range of data.

Our experiments show that $Ghap(G)$ is often smaller than $MinHK(G)$ or $MinCC(G)$ when $n < m$ and when the recombination rate is low. However, when n increases, $Ghap(G)$ becomes higher than $MinHK(G)$ or $MinCC(G)$. Our simulation shows that for dataset with 20 genotypes and 20 sites, $Ghap(G)$ is larger than $MinHK(G)$ or $MinCC(G)$ for over 80% of the data. As an example, a real biological data (from Orzack, et al. [31]) has 80 rows and 9 sites. $MinHK(G) = MinCC(G) = 2$, while $Ghap(G)$ is 5 (which is equal to $R_{min}(G)$ as shown in Section 5.3).

4. CONSTRUCTING A MINIMUM ARG FOR GENOTYPE DATA USING BRANCH AND BOUND

In this section, we consider the problem of constructing an ancestral recombination graph (ARG) that derives an HI solution H and uses the fewest number of recombinations for the genotype matrix G. We call such an ARG a minimum ARG for G and denote the minimum number of recombination in this ARG $Rmin(G)$. Formally,

Haplotyping on a minimum ARG: Given a genotype data G, find an HI solution H for G, such that we can derive H on an ARG with the fewest number of recombinations. Here, as usual, we assume the infinite sites model of mutations.

It is easy to see this problem is difficult. After all, there is no known efficient algorithm for constructing the minimum ARG for *haplotype* data [37, 3] and haplotype data can be considered to be a subset of genotype data. Here, we show that under certain conditions, we can solve this problem by a branch and bound method. The intuition of our method comes from the concept of hypercube of length m binary sequences.

Note that there are up to 2^m possible sequences in the hypercube that can be on the an ARG that derives an HI solution for G. Conceptually we can build the ARG as follows. We start from every sequence node in the hypercube as the root of the ARG. Each time, we try all possible ways of deriving a new sequence by (1) an (unused) mutation from a derived sequence, or (2) a recombination of two derived sequences. The ARG grows when we derive new sequences. Once the ARG derives an HI solution for G, we have found an ARG that is potentially the solution. We can find the minimum ARG by searching through all possible ways of deriving new sequences and finding the ARG with smallest number of recombinations.

Directly applying the above idea is not practical when the data size increases. We develop a practical method using branch and bound. We start building the ARG by staring from a sequence as the root. At each step, we maintain a set of sequences that have been derived. We also maintain the best ARG found so far, i.e. the ARG that derives an HI solution for G and use the smallest number of recombinations (denoted $Rmin$). We derive a new sequence by a recombination of two already derived sequences or an unused mutation from a derived sequence. We check whether the current ARG derives an HI solution. If so, we store this solution if this ARG uses less recombinations than $Rmin$. If not, we compute a lower bound on the minimum number of recombinations we need to derive an HI solution, given the choices we make in the search path. If the lower bound is not smaller than $Rmin$, we know the current partially built ARG can not lead to a better solution and thus stop this search path. Otherwise, we continue to derive more sequences from the current derived sequences. We illustrate the basic procedure of the branch and bound method in Algorithm GenoMinARG.

Algorithm *GenoMinARG*

1. Root We maintain a set of sequences called derived set (containing sequences that are part of the ARG already built so far). Initialize the derived set with a binary sequence s_r as the root of the ARG. Maintain a variable *Rmin* as the currently known minimum number of recombinations. Initialize *Rmin* to be ∞ (or some pre-computed upper bound).

2. Deriving sequences Repeat until all search paths are explored or terminated. Then Return to Step 1 if there are more root sequences to try. Stop the algorithm otherwise.

2.1 Through either a recombination or (unused) mutation from sequences in the derived set, grow the derived set by deriving a new sequence.

2.2 Check whether the derived set contains an HI solution. If so, stop this search path. Denote the number of recombinations in this ARG $Rmin_c$. If $Rmin_c < Rmin$, set $Rmin \leftarrow Rmin_c$. Continue with the next branch.

2.3 If the recombination lower bound (with the current derived haplotypes) is at least *Rmin*, stop this search path and continue with the next search. Otherwise, follow this branch and continue on step 2.1.

The key to the success of branch and bound is the use of genotype recombination lower bounds we presented earlier. We use the lower bound methods in Section 3 and also improve them with some additional ideas, which speed up the method significantly. We omit the details due to the space limit.

Remarks. The branch and bound method seems to work for many datasets with up to 8 sites. This method is still useful because there are real biological data that contain small number of sites and many rows. We provide such an example in Section 5.3.

5. APPLICATIONS

5.1. Detecting recombination hotspots using genotype data

Recombination rates are often believed to vary significantly across a genome. A recombination hotspot refers to a genomic region where the recombination rate is much higher than in its neighboring regions. Detecting recombination hotspots is important for many applications, e.g. association mapping and has been actively studied recently, for example in Myers, et al. [29]. Bafna and Bansal [2] applied recombination lower bounds based on *haplotypes* to reveal recombination hotspots. Their results on the MHC data [20] and MS32 data [21] indicate that recombination lower bounds may be useful in identifying locations and intensity of recombination hotspots.

However, computing recombination lower bounds on haplotype data has a potential problem. The real biological data (such as the MHC [20] and MS32 data [21]) are *genotypes*. It is not clear what effect the haplotyping error has on recombination hotspot detection. Here, we compute the *exact* minimum number of recombinations for small intervals of the *genotype* data, and thus effectively remove the haplotyping uncertainty from our results.

Given a set of genotypes G, we move a sliding window with a small number of (say 6) SNPs. We denote the submatrix within a window by G'. For each window, we compute $Rmin(G')$. Each time, we move the window by half of its width. We use $Rmin(G')$ on these intervals to calculate the average minimum number of recombinations per kB along the genome. We first analyze the MHC data. After removing missing values, there are 277 (out of 296) SNPs and 50 genotypes. On a Pentium 2.0 GHz machine, the computation takes 242 seconds when the window size is 5, and it takes 2865 seconds when the sliding window size is set to 6. For a few intervals, the computation of the exact minimum number of recombinations is slow. We simply time out and use an efficiently computed recombination lower bounds instead for these intervals.

Figure 1 plots the average minimum number of recombinations per KB across the region of interest. The results we obtained by computing over genotypes match the results over haplotype data quite

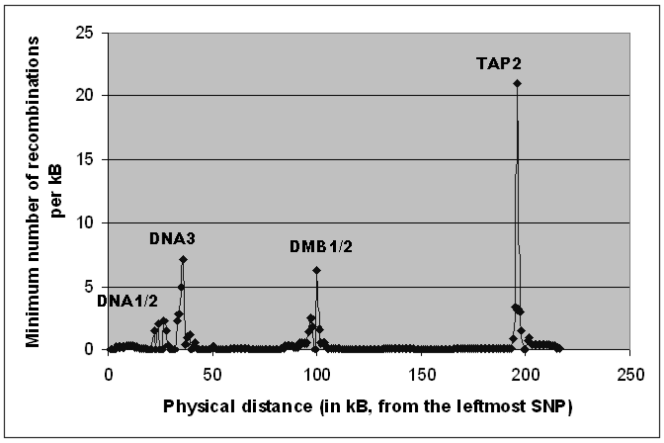

Fig. 1. Detecting recombination hotspots using genotype data for MHC data. Use a sliding window of 6 SNPs. We assign position 0 to the leftmost SNP. The results match the ones based on haplotypes.

well. The DNA1/2, DNA3, DMB1/2 and TAP2 hotspots are clearly identifiable and match the results in Bafna, et al. [2] quite well. We also tested our method on the MS32 data [21]. The result is shown in Figure 2. Again, we see good matches with the results reported in Bafna, et al. [2]. Five hotspots are identifiable: NID2, NID1, MS32, MSTM1 and MSTM2. As in Bafna, et al. [2], NID3 is not significant and not detected. Overall, our results by computing minimum number of recombination on genotype data match quite well with the results in Bafna, et al [2].

5.2. Comparing minimum recombination on genotypes and haplotypes

In Section 4, we demonstrated that for certain genotype data, we can build a minimum ARG that derives an HI solution explaining the given genotypes. This allows us to compare the minimum recombination on genotypes, original haplotypes and HI solutions found by program PHASE.

We generated simulation genotype data as follows. We first run Hudson's program MS [18] to generate $2n$ haplotypes (denoted H_o). We choose various scaled recombination rate (denoted ρ) when running MS. Genotype matrix G with n rows is then generated by pairing haplotype $2i$ with haplotype $2i-1$. We fix the number of sites in the sequences to be a small number, say 7. We run our method on G, and obtain an HI solution H. We compare H with the original haplotypes H_o. As a comparison, we run program PHASE on G, and compare the PHASE HI solution (denoted H_p) with H_o. For each data size and scaled recombination rate, we generate 100 datasets.

For each dataset, we compute $Rmin(G)$, the minimum number of recombinations on G. We also compute $Rmin(H_o)$ using program *beagle* [24]. As a comparison, we compute $Rmin(H_p)$. Note that $Rmin(G) \leq Rmin(H_o)$ and $Rmin(G) \leq Rmin(H_p)$.

Table 1 shows the comparison among $Rmin(G), Rmin(H_o)$ and $Rmin(H_p)$ for various data size and ρ. One can see that for a large portion of the data we simulated, $Rmin(G) = Rmin(H_o)$. Thus, haplotyping on a minimum ARG may be an effective approach for the range of data we tested. Another interesting observation is on the performance of PHASE. PHASE is known to be quite accurate for many data. Our simulation here shows that PHASE tends to minimize the number of recombinations to some extent, at least implicitly. Our simulation results show that often, but not always, $Rmin(H_p)$ is between $Rmin(G)$ and $Rmin(H_o)$.

From the simulation we performed, we observed that for some data, $Rmin(H_o)$ is much closer to $Rmin(G)$ than $Rmin(H_p)$. But on average, $Rmin(H_o)$ is usually closer to $Rmin(H_p)$ than $Rmin(G)$. Overall, these experiments suggest that for the downstream question of computing the minimum number of recombination, the two stage approach by first using PHASE to obtain an HI solu-

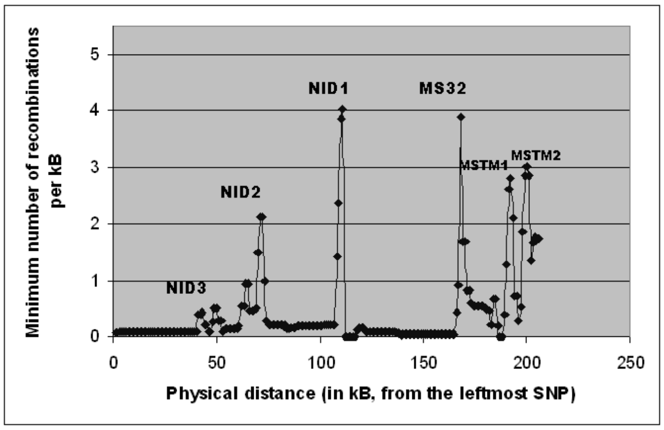

Fig. 2. Detecting recombination hotspots using genotype data for MS32 data. Use a sliding window of 6 SNPs.

Table 1. Comparison of $Rmin(H_o)$, $Rmin(G)$ and $Rmin(H_p)$. Here, H stands for $Rmin(H_o)$, G stands for $Rmin(G)$ and P stands for $Rmin(H_p)$. The data size is the number of genotype rows by the number of sites. ρ is the scaled recombination rate in Hudson' program MS. The next 3 columns display the percentage of datasets where two compared minimum number of recombinations are equal. For example, for data 15 by 7, $\rho = 40$, 43% of data have equal $Rmin(G)$ and $Rmin(H_o)$. The two columns on the right shows the average difference between $Rmin(G)$ (resp. $Rmin(H_p)$) with $Rmin(H_o)$. The value reported for difference are average differences for the two numbers over the 100 datasets.

Data size	ρ	G, H	P, H	G, P	$H - G$	$H - P$
15 by 7	20	56%	68%	63%	0.62	0.45
20 by 7	20	45%	60%	72%	0.68	0.34
15 by 7	30	38%	60%	56%	0.99	0.32
20 by 7	30	44%	56%	68%	0.79	0.34
15 by 7	40	43%	46%	54%	0.87	0.2
20 by 7	40	46%	47%	76%	0.80	0.48

tion H_p and then compute $Rmin(H_p)$ is an effective approach for the range of data we tested. We want to point out however that this conclusion would not be possible without our method of computing $Rmin(G)$, thus allowing one to study this issue.

5.3. Haplotyping on a minimum ARG

Since we can build a minimum ARG for the input genotypes, we can construct a HI solution from this minimum ARG, since the ARG gives a set of haplotypes that can explain the genotypes. Naturally, we want to study the haplotyping accuracy of this minimum-ARG method. Here, we use the same simulated data in 5.2. Table 2 shows the haplotyping accuracy of our method, and table 3 shows the result on these dataset of program PHASE. We use the following three haplotyping accuracy measures: (a) The standard error [35] is the percentage of incorrectly phased genotypes, (b) the switching error [23] is related to the incorrectly phased neighboring heterozygotes, and (c) the percentage of mis-phased 2s. The results show that our method is comparable to the solutions by PHASE in accuracy. Sometimes, our method produces an HI solution better than PHASE, although statistically PHASE is still slightly more accurate for the range of data we tested. This indicates that finding a single minimum ARG that derives an HI solution may not be enough to produce more accurate HI solutions than PHASE. Finding an ARG, either a minimum one or a near-minimum one, that gives the best haplotyping accuracy remains an interesting research problem.

Table 2. Accuracy of haplotyping on a minimum ARG. The results are averaged over 100 datasets for each parameter settings. The accuracy measures includes standard error, switch accuracy and % of mis-phased 2s. One can see that, comparing to PHASE, the min-ARG approach is comparable but underperform slightly.

min-ARG	$\rho = 20$		$\rho = 30$		$\rho = 40$	
	15x7	20x7	15x7	20x7	15x7	20x7
Std. err.	0.269	0.237	0.336	0.276	0.381	0.333
Switch	0.801	0.821	0.750	0.799	0.721	0.752
% of mis-2	10.98	10.00	14.11	11.63	15.90	13.9

Table 3. Accuracy of program PHASE on the same datasets. The results are averaged over 100 datasets for each parameter settings.

	$\rho = 20$		$\rho = 30$		$\rho = 40$	
	15x7	20x7	15x7	20x7	15x7	20x7
Std. err.	0.256	0.214	0.297	0.265	0.349	0.281
Switch	0.804	0.834	0.787	0.811	0.753	0.793
% of mis-2	10.65	8.98	12.13	11.01	14.49	11.65

Finally, we test our method with the APOE data from Orzack, et al [31]. This data has 47 non-trivial genotypes (i.e. the genotype contains more than one 2) and 9 sites. This genotype data has a real solution (i.e. experimentally determined phases). Program PHASE produces a solution within 16 seconds with 4 incorrectly phased genotypes. Our method takes about 2 minute to find an HI solution with 5 incorrectly phased genotypes, performing slightly worse than PHASE. A benefit of our method is that it computes the minimum number of recombinations for the given genotype data (under the infinite sites model). For this data, $Rmin(G) = 5$. We also note that $Rmin(H_p) = 6$ and for the real solution, $Rmin(H_o) = 7$. This is a small indication that haplotyping on a minimum (or near-minimum) ARG may be useful, i.e. real data may be derived on a minimum or near-minimum ARG.

Acknowledgments. The authors wish to thank Yun S. Song for sharing simulation scripts. Work supported by grants CCF-0515278 and IIS-0513910 from National Science Foundation.

References

1. V. Bafna and V. Bansal. The number of recombination events in a sample history: conflict graph and lower bounds. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1:78–90, 2004.
2. V. Bafna and V. Bansal. Improved recombination lower bounds for haplotype data. In *Proceedings of RECOMB 2005, LNBI Vol. 3500*, pages 569–584. Springer, 2005.
3. M. Bordewich and C. Semple. On the computational complexity of the rooted subtree prune and regraft distance. *Annals of Combintorics*, 8:409–423, 2004.
4. D. Brown and I. Harrower. A new integer programming formulation for the pure parsimony problem in haplotype analysis. In *Proc. of 2004 Workshop on Algorithms in Bioinformatics*, Berlin, Germany, 2004. Springer-Verlag LNCS.
5. D. Brown and I. Harrower. A new formulation for haplotype inference by pure parsimony. report cs-2005-03. Technical report, University of Waterloo, School of Computer Science, 2005.
6. A. Clark. The role of haplotypes in candidate gene studies. *Genetic Epidemiology*, 27:321–333, 2004.
7. I. H. Consortium. The HapMap project. *Nature*, 426:789–796, 2003.
8. I. H. Consortium. A haplotype map of the human genome. *Nature*, 437:1299–1320, 2005.
9. Z. Ding, V. Filkov, and D. Gusfield. A linear-time algorithm for the perfect phylogeny haplotyping problem. In *Proceedings of RECOMB 2005, LNBI Vol. 3500*, pages 585–600. Springer, 2005.
10. D. Gusfield. Haplotype inference by pure parsimony. In R. Baeza-Yates, E. Chavez, and M. Chrochemore, editors, *14th Annual Symposium on Combinatorial Pattern Matching (CPM'03)*, volume 2676 of *Springer LNCS*, pages 144–155, 2003.
11. D. Gusfield. Optimal, efficient reconstruction of Root-Unknown phylogenetic networks with constrained and structured recombination. *JCSS*, 70:381–398, 2005.
12. D. Gusfield, S. Eddhu, and C. Langley. Optimal, efficient reconstruction of phylogenetic networks with constrained recombination. *J. Bioinformatics and Computational Biology*, 2(1):173–213, 2004.
13. D. Gusfield, D. Hickerson, and S. Eddhu. An efficiently-computed lower bound on the number of recombinations in phylogenetic networks: Theory and empirical study. To appear in Discrete Applied Math, Special issue on Computational Biology.
14. E. Halperin, G. Kimmel, and R. Shamir. Tag snp selection in genotype data for maximizing snp prediction accuracy. *Bioinformatics*, 21:i195–i203, 2005. Bioinformatics Suppl. 1, Proceedings of ISMB 2005.
15. J. Hein, M. Schierup, and C. Wiuf. *Gene Genealogies, Variation and Evolution: A primer in coalescent theory.* Oxford University Press, UK, 2005.
16. D. Hinds, L. Stuve, G. Nilsen, E. Halperin, E. Eskin, D. Gallinger, K. Frazer, and D. Cox. Whole-genome patterns of common DNA variation in three human populations. *Science*, 307:1072–1079, 2005.
17. E. Hubbel. Personal Communication, August 2000.
18. R. Hudson. Generating samples under the Wright-Fisher neutral model of genetic variation. *Bioinfor-*

matics, 18(2):337–338, 2002.

19. R. Hudson and N. Kaplan. Statistical properties of the number of recombination events in the history of a sample of DNA sequences. *Genetics*, 111:147–164, 1985.
20. A. Jeffreys, L. Kauppi, and R. Neumann. Intensely punctate meiotic recombination in the class ii region of the major histocompatibility complex. *Nature Genetics*, 29:217–222, 2001.
21. A. Jeffreys, R. Neumann, M. Panayi, S. Myers, and P. Donnelly. Human recombination hot spots hidden in regions of strong marker association. *Nature Genetics*, 37:601–606, 2005.
22. G. Lancia, C. Pinotti, and R. Rizzi. Haplotyping populations by pure parsimony: Complexity, exact and approximation algorithms. *INFORMS J. on Computing, special issue on Computational Biology*, 16:348–359, 2004.
23. S. Lin, D. Cutler, M. Zwick, and A. Chakravarti. Haplotype inference in random population samples. *Am. J. of Hum. Genet.*, 71:1129–1137, 2003.
24. R. Lyngso, Y. Song, and J. Hein. Minimum recombination histories by branch and bound. In *Proceedings of Workshop on Algorithm of Bioinformatics (WABI) 2005*, volume 3692, pages 239–250.
25. J. Marchini, D. Cutler, N. Patterson, M. Stephens, E. Eskin, E. Halperin, S. Lin, Z. Qin, H. Munro, G. Abecasis, and P. Donnelly. A comparison of phasing algorithms for trios and unrelated individuals. *Am. J. Human Genetics*, 78:437–450, 2006.
26. G. McVean, S. Myers, S. Hunt, P. Deloukas, D. Bentley, and P. Donnelly. The fine-scale structure of recombination rate variation in the human genome. *Science*, 304:581–584, 2004.
27. A. Morris, J. Whittaker, and D. Balding. Little loss of information due to unknown phase for fine-scal linkage-disequilibrium mapping with single-nucleotide-polymorphism genotyep data. *Am. J. Human Genetics*, 74:945–953, 2004.
28. R. Morris and N. Kaplan. On the advantage of haployype analysis in the presence of multiple disease susceptibility alleles. *Genetic Epidemiology*, 23:221–233, 2002.
29. S. Myers, L. Bottolo, C. Freeman, G. McVean, and P. Donnelly. A fine-scale map of recombination rates and hotspots across the human genome. *Science*, 310:321–324, 2005.
30. S. R. Myers and R. C. Griffiths. Bounds on the minimum number of recombination events in a sample history. *Genetics*, 163:375–394, 2003.
31. S. Orzack, D. Gusfield, , J. Olson, S. Nesbitt, L. Subrahmanyan, and J. V. P. Stanton. Analysis and exploration of the use of rule-based algorithms and consensus methods for the inferral of haplotypes. *Genetics*, 165:915–928, 2003.
32. R. V. Satya and A. Mukherjee. An optimal algorithm for perfect phylogeny haplotyping. In *Proceedings of 4th CSB Bioinformatics Conference*, Los Alamitos, CA, 2005. IEEE Press.
33. P. Scheet and M. Stephens. A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. *Am. J. Human Genetics*, 78:629–644, 2006.
34. Y. Song, Y. Wu, and D. Gusfield. Efficient computation of close lower and upper bounds on the minimum number of needed recombinations in the evolution of biological sequences. *Bioinformatics*, 21:i413–i422, 2005. Bioinformatics Suppl. 1, Proceedings of ISMB 2005.
35. M. Stephens, N. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *Am. J. Human Genetics*, 68:978–989, 2001.
36. B. Voight, S. Kudaravalli, X. Wen, and J. Pritchard. A map of recent positive selection in the human genome. *PLOS Biology*, 4:e72, 2006.
37. L. Wang, K. Zhang, and L. Zhang. Perfect phylogenetic networks with recombination. *Journal of Computational Biology*, 8:69–78, 2001.
38. C. Wiuf. Inference of recombination and block structure using unphased data. *Genetics*, 166:537–545, 2004.