JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 15, Number 9, 2008

Pp. 1133–1153
DOI: 10.1089/cmb.2008.0065

Estimating the Ancestral Recombinations Graph (ARG) as Compatible Networks of SNP Patterns

LAXMI PARIDA,[1] MARTA MELÉ,[1,2] FRANCESC CALAFELL,[2]
JAUME BERTRANPETIT,[2] and THE GENOGRAPHIC CONSORTIUM[3]

ABSTRACT

Traditionally nonrecombinant genome, i.e., mtDNA or Y chromosome, has been used for phylogeography, notably for ease of analysis. The topology of the phylogeny structure in this case is an acyclic graph, which is often a tree, is easy to comprehend and is somewhat easy to infer. However, recombination is an undeniable genetic fact for most part of the genome. Driven by the need for a more complete analysis, we address the problem of estimating the ancestral recombination graph (ARG) from a collection of extant sequences. We exploit the coherence that is observed in the human haplotypes as patterns and present a network model of patterns to reconstruct the ARG. We test our model on simulations that closely mimic the observed haplotypes and observe promising results.

Key words: gene clusters, genomic rearrangements, strings, recombinations, ARG.

1. INTRODUCTION

IN THE POSTGENOMIC ERA, one of the most challenging problems is understanding genome diversity, both in its extent and in the processes that have shaped it in time and space. The differences among

[1]Computational Biology Center, IBM TJ Watson Research, Yorktown Heights, New York.
[2]Biologia Evolutiva, Universitat Pompeu Fabra, Barcelona, Catalonia, Spain.
[3]Theodore G. Schurr, University of Pennsylvania, Philadelphia, Pennsylvania; Fabrício R. Santos, Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais, Brazil; Lluis Quintana-Murci, Institut Pasteur, Institut Pasteur, Paris, France; Jaume Bertranpetit, Universitat Pompeu Fabra, Barcelona, Catalonia, Spain; David Comas, Universitat Pompeu Fabra, Barcelona, Catalonia, Spain; Chris Tyler-Smith, The Wellcome Trust Sanger Institute, Hinxton, United Kingdom; Pierre A. Zalloua, Lebanese American University, Chouran, Beirut, Lebanon; Elena Balanovska, Russian Academy of Medical Sciences, Moscow, Russia; Oleg Balanovsky, Russian Academy of Medical Sciences, Moscow, Russia; Doron M. Behar, Genomics Research Center, Family Tree DNA, Houston, Texas; R. John Mitchell, La Trobe University, Melbourne, Victoria, Australia; Li Jin, Fudan University, Shanghai, China; Himla Soodyall, National Health Laboratory Service, Johannesburg, South Africa; Ramasamy Pitchappan, Madurai Kamaraj University, Madurai, Tamil Nadu, India; Alan Cooper, University of Adelaide, South Australia, Australia; Ajay K. Royyuru, IBM T.J. Watson Research Center, Yorktown Heights, New York; Laxmi Parida, IBM T.J. Watson Research Center, Yorktown Heights, New York; Saharon Rosset, Tel Aviv University, Tel Aviv, Israel; Jason Blue-Smith, National Geographic Society, Washington, D.C.; David F. Soria Hernanz, National Geographic Society, Washington, D.C.; R. Spencer Wells, National Geographic Society, Washington, D.C.

genomes should be framed in their context of the tempo and mode of the dynamic process. In general, most of the efforts with DNA sequence have been devoted to the process of mutation, in which new variants arise with mutations and enter the dynamics driven by stochastic processes, dependent on the specific demography of the population where they have appeared, and on natural selection. At the very end, the variation that may have been produced from a single DNA sequence is an evolutionary process that, in most cases, can be reconstructed only through the knowledge of present genome variations and a model that incorporates mutations. Moreover, under a model with only mutations, given an extent and structure of genome variations, the evolutionary process of the organisms (different individuals within a species or different species) carrying them through time can be reconstructed: this is fundamental in molecular evolution.

When human history has been reconstructed through our genome two portions of it have been mostly used: mitochondrial DNA (mtDNA) and the nonrecombinant part of the Y-chromosome (Behar et al., 2007). The reason for restricting much of the evolutionary analysis to these two genetic loci is straightforward: their lack of recombination allows to reconstruct the gene tree of the extant variation in a very accurate way. This is how the ancestry of the male- and female-transmitted lineages has been traced back to a single genetic origin in individuals living in Africa in a time period around 100,000–150,000 years ago. The fundamental point in these cases is that mutations have accumulated during the human expansion out of Africa and in the settlement of continents and regions, thus allowing to locate geographically specific branches of the gene tree recognized by specific substitutions: this is the basis for phylogeography.

Most of the genome has not been amenable to accurate phylogeographic analysis, hence the analysis has been based on two single-loci: this is a major issue with the analysis. Moreover, the variance in coalescent process is very large. Thus, a sample of just two loci is woefully inadequate to reconstruct with any precision the time-depth and pattern of human evolution. To complicate matters further, if natural selection had acted in any gene of either mtDNA or the Y chromosome, the coalescence would have shrunk (for positive selection) or stretched (for balancing selection), along the whole non-recombinant Y or mtDNA, thus biasing time-depth estimates. Such local genomic effects can be compensated if multiple independent loci are used. Though such an analysis faces a particular hurdle: recombination has acted in most of the genome to create sequence variation, and reliable inferential models that incorporate recombinations are scarce. This is the aim of the present work: detecting recombination with the ultimate goal of using these recombination events as phylogeographic markers.

Barring structural variation, such as that produced by differences in copy number, sequence diversity is ultimately generated by mutations (or substitutions) and recombinations. Upon that basic material, demography will stochastically remodel the extant variations, but independent of the mutations and recombinations. Mutations and recombinations are not equivalent in their ability to be detected. While, at short time spans such as that of human evolution, each mutation that remains at moderate frequencies in a population can be sampled and identified, recombination events can superimpose on each other, thus burying the more ancient ones. Moreover, ancient recombinations may have happened in a context of low sequence diversity, giving rise to recombinant haplotypes that are virtually indistinguishable from their parental sequences. It follows, then, recombination detection will perform best if confronted with the most recent events.

Mutation creates alternating states at particular genome positions known as single nucleotide polymorphisms (SNPs); a genome sequence can be reduced to a set of SNPs, and recombination will shuffle these sequences to produce new haplotypes. The coalescence time of a SNP is a direct function of its allele frequencies (Watterson and Guess, 1977); if only those SNPs with non-extreme frequencies are considered (say, for example, $0.2 < q < 0.8$ or even $0.1 < q < 0.9$), recent mutation events will be filtered out, and most recent variation would have been produced by recombination. In other words: choosing SNPs with non-extreme allele frequency (which, on the other hand, are those mostly represented in commercial genotyping arrays) recombination may be taken as the most important factor having shaped the genetic variation in a set of contiguous SNPs that have a much older age in the gene genealogy, and will be easier to detect.

Differences in genetic variations along a chromosome in different human populations indicate population-specific recombinations and these can be used as phylogeographic markers. Its possible application to the analysis of origins in humans then becomes a powerful tool if the computational model gives a robust estimation of the recombinations based on the consecutive SNP variants in a DNA region.

Historically, the computational methods to study recombinations has been through the use of *linkage disequilibrium*. In each of the studies below, it was demonstrated that the human genome can be segmented into *haplotype blocks*, sizable regions over which there is little evidence for historical recombination. In the seminal papers (Daly et al., 2001; Patil et al., 2001), the authors showed a striking lack of diversity in chromosome 5 and chromosome 21, respectively.

In Gabriel et al. (2002), the authors characterize haplotype patterns across 51 autosomal regions (spanning 13 megabases of the human genome) in samples from Africa, Europe, and Asia. In Wang et al. (2002), the authors analyze haplotype data on chromosome 21 focusing on recombinations between haplotype blocks to study the combined effects of demographic history and various population genetic parameters on haplotype block characteristics. The full analysis of the HapMap data in its I and II phases (International HapMap Consortium, 2007) has given a full detailed description of linkage disequilibrium and population recombination rates among every pair of SNPs at high density.

Broadly speaking there have been two genetic models, proposed in literature. The first is the *block model*. The authors in Zhang et al. (2002) develop a dynamic programming algorithm for haplotype block partitioning to minimize the number of representative SNPs required to account for most of the common haplotypes in each block. Yet another discrete method to study the block model has been termed the *mosaic model* (Mannila et al., 2003; Ukkonen, 2002; Wu and Gusfield, 2007). Here the assumption is that the current population evolved from a small set of founder sequences and the extant sequences are a mosaic of these founder sequences. One of the optimizing criteria is to minimize the number of founder sequences or minimizing the fragmentation of the extant sequences. The advantage of the mosaic model, over the block model, is that it can identify the recombinant and the ancestor sequences. However the ancestor sequences are always parts of the founder sequences, whose inference has been an unsolved challenge. Also, authors in Song et al. (2006) count all possible ancestral combinations for a given collection of haplotypes using a dynamic programming.

The second is the *phylogenetic model* which is the inclusion of the recombination events in a mutation based phylogeny; the resulting topology is a directed acyclic graph (DAG), also called a *phylogenetic network* (Gusfield et al., 2007; Wu and Gusfield, 2007; Moret et al., 2004). We discuss one such model here: The original (or ancestral) state of each SNP is specified and encoded as 0 and the changed state is encoded as 1. This model assumes a single ancestor (or founder sequence) encoded as a series of 0's. Each internal node encodes the intermediate sequence in the evolution process and the extant sequences appear on the leafnode of the DAG. A recombination is modeled as a "hybrid" of two parental sequences where at each position it has the value from one of the parental sequences. The mosaic model has the advantage of naturally allowing for multiple founder sequences. While this model cannot detect ancient recombinations (i.e., intermediate recombinant sequences), the phylogenetic model has the potential for recovering the intermediate states. The phylogenetic model is very clean and elegant, and some interesting theoretical results (on the network structure) are presented for one such model in Gusfield et al. (2007). Also, heuristic based algorithms have been used to infer plausible ancestral recombinations graph (ARG) for a collection of haplotypes (Minichiello and Durbin, 2006).

2. OUR MODEL: MINIMAL COMPATIBLE NETWORK

We propose a model that exploits the contiguous patterns in the extant sequences, as in the mosaic model, to construct a phylogenetic network. A striking difference from the other network models is that ours permits multiple roots (or founder sequences). Further, a recombination is modeled as a hybrid of stretches (or *segments*) of parental sequences. We do this by extending the notion of *compatible trees* to networks that models recombinations in a fairly generalized form. The network is defined in terms of a segmentation S of the aligned extant sequences into say K segments. Every edge in the network is associated with at least one segment (from the K segments). When the network is restricted only to the edges of fixed segment, the resulting topology is a phylogenetic tree of only that segment from all the extant sequences.

We formally define the compatible network here. Let I be the given input matrix with n rows and m columns. Each row corresponds to an extant sequence or haplotype. The matrix I has two kinds of elements, *solid characters* and dont-care. As our model will not only consider single character (where it

would be normal to use the four nucleotides A, C, G, T) but combinations of them (see *grain* below) the nomenclature is not restricted to four states. A dont-care is written as '-' and its semantics will be discussed later. Further, for ease of exposition, let all occurrences of a character c be within a single column of I. See Figure 1 (1) for an example.

The *segmentation* S of the m columns of input I, written as the closed interval $[1, m]$, is a collection of non-overlapping intervals such that each column j is in at most one segment. For example, given 5 columns, a possible segmentation of $[1, 5]$ is:

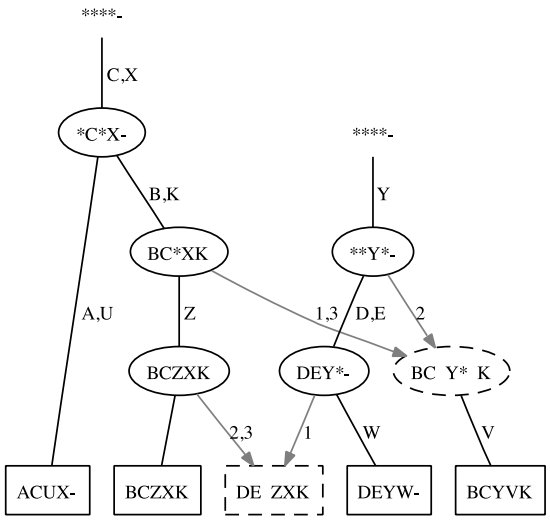
$$S = \{[1, 2], [3, 4], [5, 5]\}$$

For convenience, the three segments are denoted simply by integer labels 1, 2, and 3.

A *compatible network* N is a directed acyclic graph (DAG) that explains input I with a segmentation S. N is defined as follows: It has three kinds of nodes. A node with no incoming edge is a *root* node and N may have multiple root nodes. A node with no outgoing edges is a *leaf* node and there can be no more than n leaf nodes in N. Every other node is an *internal* node. An internal node has at most two incoming edges. When a node has exactly one incoming edge, it is called a *mutation node* and the incoming edge is called a *mutation edge*. When the node has two incoming edges, the node is called a *recombination* or a *hybrid* node and the incoming edges are called *recombination edges*. A mutation edge is labeled with element(s) from the matrix I. A recombination edge is labeled with segment(s) of S.

Each node in N is labeled by a sequence of length m. Let e be the mutation edge coming into node v with labels $c_1, c_2, \ldots c_l$. Let c_i occur in column j_i of I. Then the label of v is obtained from the label of its only parent, by replacing positions $j_1, j_2, \ldots, j_l$ with values $c_1, c_2, \ldots, c_l$ respectively.

Let e_1 and e_2 be two recombination edges coming into a node v with segment labels $s_1, s_2, \ldots s_{l_1}$ and $r_1, r_2, \ldots r_{l_2}$ of parents 1 and 2, respectively. The label of node v is a hybrid sequence with segments $s_1, s_2, \ldots s_{l_1}$ from the label of the first parent and segments $r_1, r_2, \ldots r_{l_2}$ from the label of the second parent. Each position in the missing segments, i.e., neither from parent 1 nor from parent 2, in the label is written as ϕ. The interpretation of ϕ is that it is a sort of a filler, that is not reflected in any way in the extant sequences and could be ignored for all practical purposes (for the interested reader, such a node whose



A	C	U	X	-
B	C	Z	X	K
D	E	Z	X	K
D	E	Y	W	-
B	C	Y	V	K
segment labels	1	1	2	2	3

(1) Input matrix I and the three segments with labels 1,2,3.

(2) A compatible network N.

FIG. 1. **(1)** The input 5×5 matrix I where, for convenience, distinct characters occur in each column. The segmentation of the five columns is $S = \{[1, 2], [3, 4], [5, 5]\}$ with the three segment labels as shown. A compatible network for I with segmentation S is shown in **(2)**. The leaf nodes are shown as rectangular boxes and correspond to the five rows of I; the internal nodes are shown as ellipses and the three root nodes are shown with labels having only "*" or "-." The direction of the edges is always towards the leaf nodes which in this rendering of network N is downwards. To avoid clutter, only the recombination edges display the direction as arrows. The recombination or hybrid nodes are shown in dashed boundaries.

label must have ϕ in some position(s), is marked x in Figure 10 (3)). In genetic terms these are partial sequences that have been lost and their existence is known because only part of it (a recombinant fragment) has reached the present. An interesting example is found in the ABO sequence variations (Roubinet et al., 2004).

Next, the network N is compatible with input I with segmentation S if the two conditions hold:

(1) the label of a leaf node corresponds to a row in matrix I and every row in matrix I is the label of some leaf node in N.
(2) For each column j in I, every solid character (of j) occurs exactly once in the label of some mutation edge in N.

See Figure 1 for a concrete example. The (only) internal hybrid node is labeled BCY*K. This hybrid node has two "breakpoints": the first two columns (segment 1) and the last column (segment 3) are from one parent and the remaining middle two columns (segment 2) are from the other parent: this kind of recombination has been termed *two crossover recombination* or *gene conversion* in literature. The leaf hybrid node, labeled DEZXK, is explained as a recombination where the prefix (first two columns or segment 1) is from one parent and the suffix (last three columns or segments 2 and 3) is from the other.

2.1. The compatible network construction problem

For a segment $s \in S$, $Restricted(N, s)$ is the network obtained by doing the following two operations. (1) Removing all recombination edges that do not have the label s. (2) Let character c occur in column j and $j \notin s$, then the label c can be removed from the mutation edge label.

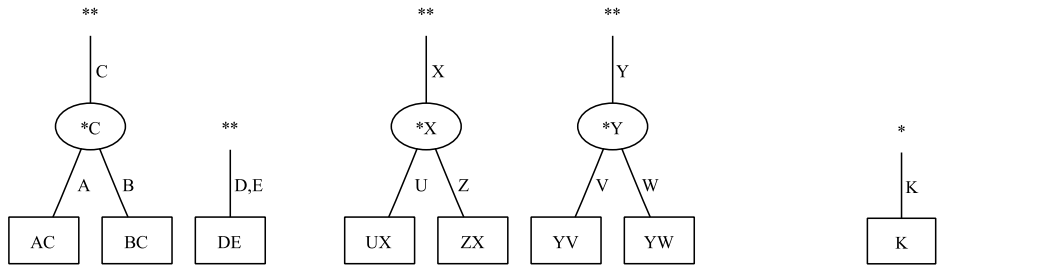
Observation 1. *For each segment $s \in S$, $Restricted(N, s)$ is a forest, i.e., each connected component is a tree.*

Figure 2 shows an example of restricted networks. $L(N, s, c)$ is the collection of rows corresponding to leafnodes reachable from node v in $Restricted(N, s)$ where v has an incoming mutation edge with the label c. Note that for a fixed c, the node v is unique in a compatible network. We pose the following optimization problems.

Problem 1 (*Minimal Segmentation). Given I, the task is to compute a compatible network N with minimum number of elements in segmentation S of I.*

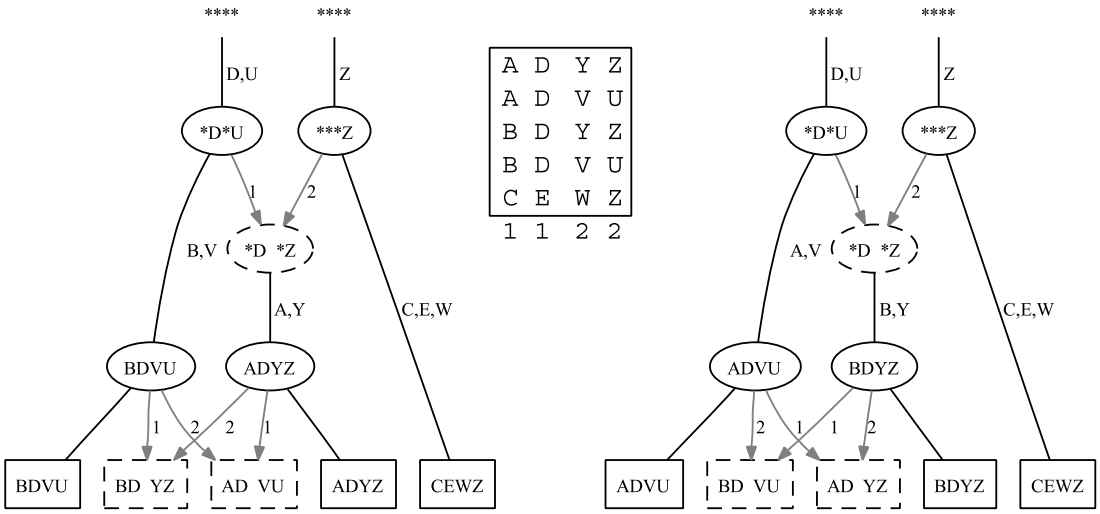
Problem 2 (*Minimal Recombination). Given I, the task is to compute a compatible network N with minimum number of recombinations in N.*

Note that the number of recombinations in a compatible network is at least $K - 1$, where K is the number of segments in the segmentation S.



(1) $Restricted(N, [1, 2])$. (2) $Restricted(N, [3, 4])$. (3) $Restricted(N, [5, 5])$.

FIG. 2. The three forests corresponding to the three segments of S in the compatible network N of Figure 1: **(1)** $Restricted(N, [1, 2])$, **(2)** $Restricted(N, [3, 4])$, **(3)** $Restricted(N, [5, 5])$.



(1) Compatible network N_1. (2) Compatible network N_2.

FIG. 3. The input 5×4 matrix I with segmentation of the 4 columns as $S = \{[1,2],[3,4]\}$. Two distinct compatible networks N_1 **(1)** and N_2 **(2)** for the same segmentation S are shown above. For notation, see Figure 1.

Observation 2 (*Nonuniqueness). Two distinct segmentations $S \neq S'$ can give distinct compatible networks N and N'. Moreover, it is possible that a segmentation S, can give two distinct compatible networks N_1 and N_2.*

Figure 3 shows a character matrix I and two distinct compatible networks for the same segmentation S (the interested reader can see an example in Figure 7, where two distinct segmentations of the same size ($K = 2$) is possible for the same character matrix).

We propose to tackle Problem 1 in a three step process: In the first step we transform the input haplotypes into a character matrix; in the second step we split this character matrix into segments where a phylogenetic forest can explain each segment and then in the final step construct the compatible network from the forests. The details follow.

3. (STEP 1) STAGING THE INPUT: HAPLOTYPES TO "CHARACTER" MATRIX

The input is a collection of haplotypes, in the form of a matrix I' where each row is an ordered vector of SNP values as they appear along a chromosome. A running example of 25 haplotypes each with 85 SNP's is shown in in Figure 4. A reference sequence is shown at the top. The asterisk in the haplotype denotes agreement with the reference sequence.

We process this data into a smaller matrix (I) of blocks of SNPs that we call *characters* (not unlike taxonomic characters) since each column may now take on multiple values and further, the order is unspecified. This processing is carried out in the following three stages.

1. Removing redundancies: Without loss of generality, no two rows are identical in I'. However, since our interest is in recombinations, identical columns are not considered redundant for the purpose of extracting the topology of the phylogenetic network. Note, however that the information regarding the *redundant* rows is retained for the final analysis.

2. Grouping the columns (grain *g*): A fixed number, g, of consecutive columns of I' are blocked together to obtain I''. Thus, each column of I'' represents a block of g consecutive SNPs from the input matrix I'. Each distinct pattern in I'' is assigned a distinct integer label. Note that for the purposes of extracting the topology, it is adequate to distinguish different patterns in a single column. Thus a label "2" in column one may represent an entirely different pattern from label "2" in column four. (Note that we used uppercase alphabets for these labels, instead of integers, in the previous section for ease of exposition.)

```
     123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345
Ref  GCCCCCACGGAGGCGGTGCCGGCGCCTGCTAGCAGACCTGAAGGGTCAGAGACCGATACGGCAGCCGCTAGCGCGCTCATGCCGG

 1   CT*G***T*A*****ACCGGTC***G****CT***************GA************************A********AA
 2   CT*G***T*A*****ACCGGTC***G*****T***************GA************************A********AA
 3   CT*G***T*A*****A*CGGTC***G*****T***************GA************************A********AA
 4   CT*G***T*A*****ACCGGTC***G*****T***************GA************************A********AA
 5   CT*G***T*A*****ACCGGTC***G****CT***************G*************************A********AA
 6   CT*G***T*A*****ACCGGTC***G*****T***************G*************************A********AA
 7   ***A******GA*****TGGCC***GC******GA***C*************T*********A******A**********AA
 8   ***A*******A*T***TGGCT**GGC*****************************************************AA
 9   ***A*******A*T***TGGCT**GGC*********************A*******************************AA
10   ***A*******A*T***TGGCT**GGC*********************A*******************************AA
11   ***A*******A*T***TGGCT**GGC***************CA****A*******************************AA
12   ***A*******A*T***TGGCT**GCC***************CA****A*******************************AA
13   ***A*******A*T***TCCCT**GCC*************************************G*****************AA
14   ***A*******A*T***TCCCT**GGC******C*************************G*****************AA
15   ***A*******A*T***TGGCTT*GGC******C*************************G***************CCC*A*
16   ***A*******A*T***TGGCT**GGC*****************************************************A*
17   ***A*******A*T***TGGCT**GGC*****************************************************AA
18   ***A*******A*T***TGGCT**GGC*****************************************************AA
19   ***A*******A*T***TGGCT**GGC***********A*******************A*****C*CG****A***G*****A
20   ***A*******A*T***TGGCT**GGC***********A*******************A*****C*CG****A***G*****A
21   ***G*******AA*A**TGGCC***GC******G*****A*******************A*****C*CG****A***G*****A
22   ***A*******A*T***TGGCC**GGCA*********C**G*****************A****TC*C*****A*********A
23   ***A*******A*T***TGGCC**GGCA**********C**G*****************A****TC*C*****A*********A
24   ***A*******A*T***TGGCC**GGCA*****CA***C**G*****************A****TC*C*****A*********A
25   ***A*******A*T***TGGCC**GGCA*****CA***C**G*****************A****TC*C*****A*********A
```

(1) I': Input matrix of SNPs.

```
     1 2 3 4 5 6 7 8 910 1 2 3 4 5 6 7 8 910 1 2 3 4 5 6 7 8 9

 1   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 2   1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 3   1 1 1 1 1 1 0 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 4   1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 5   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
 6   1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
 7   2 2 2 0 1 3 2 1 2 1 3 0 2 1 1 2 2 0 1 1 1 0 1 0 2 1 1 1 1
 8   2 2 2 3 2 3 2 2 3 1 3 1 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1 1 1
 9   2 2 2 3 2 3 2 2 3 1 3 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1
10   2 2 2 3 2 3 2 2 3 1 3 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1
11   2 2 2 3 2 3 2 2 3 1 3 1 1 1 2 2 1 1 1 1 1 1 1 1 2 1 1 1 1
12   2 2 2 3 2 3 2 2 4 1 3 1 1 1 2 2 1 1 1 1 1 1 1 1 2 1 1 1 1
13   2 2 2 3 2 3 3 2 4 1 3 1 1 1 1 2 2 1 1 1 2 1 1 1 2 1 1 1 1
14   2 2 2 3 2 3 3 2 3 1 3 3 1 1 1 2 2 1 1 1 2 1 1 1 2 1 1 1 1
15   2 2 2 3 2 3 2 0 3 1 3 3 1 1 1 2 2 1 1 1 2 1 1 1 2 1 0 0 2
16   2 2 2 3 2 3 2 2 3 1 3 1 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1 1 2
17   2 2 2 3 2 3 2 2 3 1 3 1 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1 1 1
18   2 2 2 3 2 3 2 2 3 1 3 1 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1 1 1
19   2 2 2 3 2 3 2 2 3 1 3 1 1 2 1 2 2 1 1 1 3 1 2 3 1 1 3 3 1
20   2 2 2 3 2 3 2 2 3 1 3 1 1 2 1 2 2 1 1 1 3 1 2 3 1 1 3 3 1
21   2 1 2 3 0 3 2 1 2 1 3 0 1 2 1 2 2 1 1 1 3 1 2 3 1 1 3 3 1
22   2 2 2 3 2 3 2 1 3 2 3 1 2 3 1 2 2 1 1 1 3 3 2 1 1 1 1 3 1
23   2 2 2 3 2 3 2 1 3 2 3 1 2 3 1 2 2 1 1 1 3 3 2 1 1 1 1 3 1
24   2 2 2 3 2 3 2 1 3 2 3 5 2 3 1 2 2 1 1 1 3 3 2 1 1 1 1 3 1
25   2 2 2 3 2 3 2 1 3 2 3 5 2 3 1 2 2 1 1 1 3 3 2 1 1 1 1 3 1
```

(2) I'': Blocking $g = 3$ SNPs of I'.

```
cluster  1 2 3 4 5 6 7 8 910 1 2 3 4 5 6 7 8 920 1 2 3 4 5 6 7 8 9   (rows of I'')

 1       1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1          (2 3 4)
 2       2 2 2 3 2 3 2 1 3 0 3 1 2 0 1 2 2 1 1 1 3 0 2 1 1 1 1 3 1    (22 23 24 25)
 3       2 2 2 3 2 3 2 2 3 1 3 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1          (9 10)
 4       2 2 2 3 2 3 2 2 3 1 3 1 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1 1 1        (8 17 18)
 5       2 2 2 3 2 3 2 2 3 1 3 1 1 2 1 2 2 1 1 1 3 1 2 3 1 1 3 3 1          (19 20)
 6       1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1             (1)
 7       1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1             (5)
 8       1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1             (6)
 9       2 2 2 0 1 3 2 1 2 1 3 0 2 1 1 2 2 0 1 1 1 0 1 0 2 1 1 1 1             (7)
10       2 2 2 3 2 3 2 2 3 1 3 1 1 1 2 2 1 1 1 1 1 1 1 1 2 1 1 1 1           (11)
11       2 2 2 3 2 3 2 2 4 1 3 1 1 1 2 2 1 1 1 1 1 1 1 1 2 1 1 1 1           (12)
12       2 2 2 3 2 3 3 2 4 1 3 1 1 1 1 2 2 1 1 1 2 1 1 1 2 1 1 1 1           (13)
13       2 2 2 3 2 3 3 2 3 1 3 3 1 1 1 2 2 1 1 1 2 1 1 1 2 1 1 1 1           (14)
14       2 2 2 3 2 3 2 0 3 1 3 3 1 1 1 2 2 1 1 1 2 1 1 1 2 1 0 0 2           (15)
15       2 2 2 3 2 3 2 2 3 1 3 1 1 1 1 2 2 1 1 1 1 1 1 1 2 1 1 1 2           (16)
16       2 1 2 3 0 3 2 1 2 1 3 0 1 2 1 2 2 1 1 1 3 1 2 3 1 1 3 3 1           (21)
```

(3) I: Clustering of the rows of I''.

FIG. 4. The steps involved in processing input SNP matrix I' to obtain the character matrix I''. See text for details and Figure 5 for the explanation of one of the clusters represented in I. **(1)** I': Input matrix of SNPs. **(2)** I'': Blocking $g = 3$ SNPs of I'. **(3)** I: Clustering of the rows of I''.

In Figure 4, a grain size of $g = 3$ is used on I' to give a 25×29 matrix I''. Each column represents a block of three consecutive SNPs (except the very last column which has only one SNP). Each distinct pattern is assigned a distinct integer. For example, in column 1, pattern `CTC` (or `CT*` in I') is assigned integer 1; pattern `GCC` (or `***` in I') is assigned integer 2.

Wild character `0` *in* I'. A unique pattern in a column is assigned 0. Note that a single column may have multiple 0's. For example, in Figure 4 (2), column 12 has a value 0 at rows 7 and 21. The pattern `GAA` appears only in row 7 and the pattern `GGA` appears only once in row 21 in that column.

Discussion. It is possible to treat multiple patterns as a single pattern based on various criteria. One of them could be to use edit distance between the patterns. Another may depend on fitting an evolutionary tree (no recombinations) to the g consecutive SNPs identifying potential homoplasies. Yet another possibility is to use different grain sizes along the chromosome. However, for simplicity, in this paper we use exact identity of the patterns and a single grain size for the entire extent.

What is good value of g? A good indicator is the number of wild characters (`0`) per column in I''. For a reliable analysis, this number should be low. In our experiments, we keep this number to be not more than 20% of the number of samples (or rows in I').

3. Clustering the rows: In this step, we reduce the number of rows of I'' by identifying clusters of rows that (possibly) do not have any recombinations in their evolutionary history.

Each cluster is represented by a single vector that is the putative root node of the evolutionary history (tree) of the cluster elements.

We use a greedy iterative algorithm to compute the clusters. This is based on uninterrupted patterns across the rows of the matrix and the process is summarized below.

Wrapper Procedure:

(Step 1) In each column of I'', replace a unique entry by 0.
(Step 2) Collapse I'' to $\breve{I}$, where a row in $\breve{I}$ is the putative root of a cluster of rows of I'' (see below).
(Step 3) IF I'' is identical to $\breve{I}$, terminate the process.
ELSE Update $\breve{I}$ to I'' and go to Step 1.

The pseudcode for *Collapse*, the process of clustering multiple rows into one putative root vector, is given below. This is a recursive process and the very first call is made with $c = 1$; r_{max} is the number of rows and c_{max} is the number of columns in I; and *pat* is initialized to the empty string ϕ. The concatenation of string *pat* followed by string p is written as $pat \oplus p$: this is the putative root vector that is built incrementally in the successive calls of the routine. The calls terminate either when singleton sets ($\mathcal{S}$) are reached or the rightmost column c_{max} is reached.

Collapse(I, c, c_{max},r_{max}, *pat*)

(1) IF ($c > c_{max}$) THEN rows of I is a cluster and *pat* is the putative root
(2) ELSE {
(3) $\mathcal{S}_{zero} \leftarrow \{i \mid I[i,c]$ is '0'$\}$
(4) FOR each $p \neq$ '0' in column c of I {
(5) $\mathcal{S}_p \leftarrow \{i \mid I[i,c]$ is $p\} \cup \mathcal{S}_{zero}$
(6) Restrict I only to the r $(= |\mathcal{S}_p|)$ rows of $\mathcal{S}_p$ to get $\tilde{I}$
(7) IF ($r > 1$) THEN Collapse($\tilde{I}$, $c + 1$, c_{max}, r, $pat \oplus p$)
(8) }
(9) }

The structure of the tree (plausible evolutionary history) of a cluster with the putative root is embedded in the run-time history of the wrapper procedure and the recursive calls of Collapse(). We omit the details here to avoid digression. It suffices to assume that this tree can be constructed with ease. Note that it is possible to get overlapping clusters using Collapse(), due to the use of $\mathcal{S}_{zero}$ in the procedure: see line (5) above. When a sample, say i, appears in multiple clusters, the result is to be interpreted as follows: for

	1	2	3	4	5	6	7	8	910	1	2	3	4	5	6	7	8	920	1	2	3	4	5	6	7	8	9
cluster 2 = I [2]	2	2	2	3	2	3	2	1	3	0	3	1	2	0	1	2	2	1	1	1	3	0	2	1	1	1	1	3	1
22	2	2	2	3	2	3	2	1	3	2	3	1	2	3	1	2	2	1	1	1	3	3	2	1	1	1	1	3	1
23	2	2	2	3	2	3	2	1	3	2	3	1	2	3	1	2	2	1	1	1	3	3	2	1	1	1	1	3	1
24	2	2	2	3	2	3	2	1	3	2	3	5	2	3	1	2	2	1	1	1	3	3	2	1	1	1	1	3	1
25	2	2	2	3	2	3	2	1	3	2	3	5	2	3	1	2	2	1	1	1	3	3	2	1	1	1	1	3	1

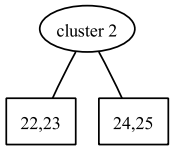


FIG. 5. Possible evolutionary history of cluster 2. The putative root node (vector) for this history is row 2 of I of Figure 4.

the granularity used, the analysis is unchanged whether i is in one cluster or the other(s). The strength of this approach is in the flexibility it provides in terms of alternative plausible hypotheses for a data set.

Figure 4 (3) shows a clustering of the samples of I''. Figure 5 describes cluster 2 from the example and shows its possible evolutionary history as a tree. Note that the four rows 22, 23, 24, 25 of I'' are identical except in column 12 where rows 22 and 23 have a value 1 and rows 24 and 25 have a value 5. Thus these two groups are split in the tree shown in Figure 5. The vector representing the putative root of this cluster has a value of 1 in column 12. This is because the value 5 does not appear in any of the remaining rows (other than rows 22, 23, 24, 25) in I' at column 12; however value 1 does appear in some of the others. Were it the case, that value 1 did not appear in any of the remaining rows, then the vector would have had a value 0 in this column.

Interpretation of wild character `0` *in* I''. This is best explained through the example of Figure 5. Consider columns 10, 14, and 22. In each of these columns, rows 22, 23, 24, and 25 of I' have an identical value of 2, 3, and 3, respectively. Also, none of the remaining rows (other than rows 22, 23, 24, 25) have this value. Hence the vector representing the putative root has the value 0 in columns 10, 14, and 22.

The value 0 in the matrix is called a *wild* character, since it can be effectively grouped (see line (5) of procedure Collapse()) with any other non-zero character, p, of that column.

4. (STEP 2) COMPUTING SEGMENTATION S

In this step, we segment the matrix I into as small a number of segments as possible, such that for each segment there exists a *compatible forest* (described in Problem 3 at the end of the section). However, we must first introduce some terminology to understand this problem setting and its proposed solution.

Let S be a segmentation with K segments of the input $n \times m$ character matrix I. For each $1 \leq k \leq K$, let the kth segment be $s^k = [j_1^k, j_2^k]$. For convenience, the size of the kth segment is $sz_k = j_2^k - j_1^k + 1$. Then I^k is the $n \times sz_k$ sub-matrix obtained by extracting the columns $j_1^k \leq j \leq j_2^k$ of I.

The segmentation S is such that for each (sub)matrix I^k there exists a compatible network N^k with no recombination nodes and a single segment in its segmentation. In other words, N^k is a forest, i.e., every connected component is a tree.

Character array I *to multisets* C*'s*. Let column j of the matrix I have L^j distinct characters, where none of the characters is a dont care ('-'). Let these characters of column j be $c_1^j, c_2^j, \ldots, c_l^j, \ldots, c_{L^j}^j$. For each distinct character c_l^j of column j, define a set of rows (or sample numbers) as follows:

$$C_l^j = \{i \mid I[i, j] \text{ is } c_l^j\}.$$

Thus each column j can be written as a multiset (or, set of sets):

$$\mathcal{C}^j = \{C_1^j, C_2^j, \ldots, C_l^j, \ldots C_{L^j}^j\}.$$

Let C_0^j is the set of `0`'s or wild cards in column j and is called the *zeroset* of column j.

Two sets C_1 and C_2 *straddle* if both the conditions hold: (1) the intersection of C_1 and C_2, $C_1 \cap C_2$, is not empty, and, (2) both the set differences $C_1 \setminus (C_1 \cap C_2)$ and $C_2 \setminus (C_1 \cap C_2)$ are non empty. As an example, let $C_1 = \{1, 2, 3\}$, $C_2 = \{2, 3, 4\}$, $C_3 = \{3, 4\}$ and $C_4 = \{5, 6\}$. Then C_4 does not straddle with any of the other sets and C_3 does not straddle with C_2. However, C_1 and C_2 straddle; C_1 and C_3 straddle.

Interplay of *wild* and *dont care* characters with segmentation *S*: Until now, we have assumed that the dont care characters are already marked in the input matrix I. The interpretation of a dont care character is that it was possibly the original state. In other words, this was the state in the root or founder sequence, hence does not require to be displayed as a mutation in the compatible network. We use the following heuristics in our approach.

(1) At most one character per column is permitted to be designated a dont care character. However, under our model, we have the flexibility of allowing $0 \leq A < m$ characters to be designated as dont cares.

Note that when each character takes binary values with $A = 1$, this condition is equivalent to Wilson's Test (Wilson, 1965) for two columns and also called the *Four Gamete Rule*. The rule states that no more than three of any of the four possibilities must be observed in the rows of the data for the two columns to be compatible.

How is the Four Gamete Rule explained using dont care characters? Figure 6 gives an explicit example, with two possible scenarios for a single data set.

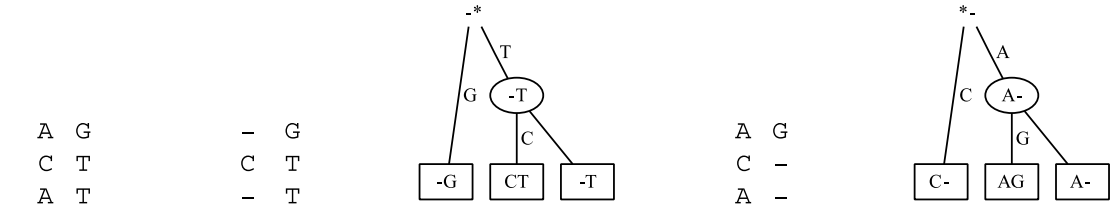
(2) Recall a character that is neither wild ("0") nor dont care ("-") is called *solid*. We impose the following condition on the compatible network. Let node v_1 have an incident mutation edge e emerging from node v_2 and edge e has a label c that occurs in column j. Let c_1 be the character in position j of the label of v_1 and c_2 is similarly defined. If $c_1 \neq c_2$ then one of the following must hold:
   (a) If c_1 is a solid character, then c_2 cannot be a solid character, i.e., it must be "∗" or "-."
   (b) If $c_1 =$ "0," then c_2 can be any character.

For a node v, let L_v denote all the leaf nodes reachable from node v in the compatible network. The following is a consequence of Heuristic (2).

Observation 3 (*Multiple roots). For each node v, there exists some position j such that for all labels in L_v, position j has some fixed value c or wild character "0." Thus it is possible to have multiple roots in the compatible network.*

We give a few definitions before the next observation. Given a segment label s, a s-path in a network is a path of edges where for every recombination edge e, s is in the label of e. Two nodes v_1 and v_2 are *incomparable* if there exists no segment label s such that there is a s-path from a root to a leaf node with both v_1 and v_2 in the path.

Observation 4 (*Incomparable mutation nodes). Let v_1 and v_2 be two mutation nodes with incoming edges e_1 and e_2 respectively with solid character labels c_1 and c_2 at column j. Then v_1 and v_2 must be incomparable in the compatible network.*



(1) Input I. (2) Designation of dont care in column 1. (3) Tree in (2). (4) Designation of dont care in column 2. (5) Tree in (4).

FIG. 6. Illustration of dont cares using the Four Gamete Rule: The input I in **(1)** has three (distinct) gametes and a tree can explain the data as follows. Two possible dont care designations in **(2)** and **(4)** and the corresponding trees are shown in **(3)** and **(5)**, respectively. A '∗' in the root node is placed in the position that does not have a dont care. Note that in general, if I (with two columns) has *no more* than three distinct gametes, a tree can explain I; however, no tree can explain I, with 3 gametes, without the use of any dont cares.

The following is a consequence of Heuristics (1) and (2).

Observation 5 (*Compatible columns with zeroset partitions). Let two columns j and k, of character matrix I, have multisets $\mathcal{C}^j$ and $\mathcal{C}^k$. Further, let the zeroset C_0^j be partitioned into some l^j sets*

$$C_{01}^j, C_{02}^j, \ldots, C_{0l^j}^j.$$

Then at least $l^j - 1$ of these augments a distinct element (set) of $\mathcal{C}^j$ to obtain a new multi-set $\mathcal{C}^{j'}$. Similarly, C_0^k is partitioned into some l^k sets $C_{01}^k, C_{02}^k, \ldots, C_{0l^k}^k$ and a new multi-set $\mathcal{C}^{k'}$ is obtained. Then columns j and k are compatible for the given $A(\geq 0)$ if

1. *at most A non-zeroset(s) can be removed from $\mathcal{C}^{j'}$ to obtain $\mathcal{C}^{j''}$ and at most A non-zeroset(s) can be removed from $\mathcal{C}^{k'}$ to obtain $\mathcal{C}^{k''}$ and,*
2. *each pair of augmented non-zerosets, one from $\mathcal{C}^{j''}$ and the other from $\mathcal{C}^{k''}$, does not straddle.*

Informally speaking, the zerosets contain wild characters which in principle can be included in any of the sets; thus a partition of the zeroset realizes this assignment to the non-zerosets of that column. In other words, this assignment handles the combinatorics due to the wild cards. The two conditions in the observation above then handle the combinatorics due to the dont care characters or the application of the generalized Four Gamete Rule. A concrete example is discussed below (after the Problem statement).

These observations lead up to a possible solution to the following problem.

Problem 3 (*Compatible Forest Problem). Given a matrix I, the task is to find if there exists some partitioning of the zeroset C_0^j of each j such that any pair of columns j and k are compatible using the partitions of the zerosets for the given A.*

It is possible to ignore the wild characters by replacing each by a non-zero (unique) integer. Further, when the character matrix I is binary and the wild characters are ignored, this problem has been called *perfect phylogeny* in literature. Even this restricted problem is known to be NP-hard (Wu and Gusfield, 2007).

We use a greedy algorithm to segment the matrix I into as small a number of segments as possible, such that for each segment there exists a compatible forest. Figure 7 shows a example of I and three possible segmentations using $A = 1$, i.e., at most one character per column is designated as a dont care. The first one has the minimum number of segmentations, i.e., 1. Nevertheless, we study two more segmentations for illustrative purposes. For partitioning of zerosets, consider the multisets for columns 5, 6, and 7 of I of Figure 7 (1):

$\mathcal{C}^5$	$\mathcal{C}^6$	$\mathcal{C}^7$
$C_0^5 = \{1\}$,	$C_0^6 = \{1,2,4,6\}$,	$C_1^7 = \{1,2,3,6\}$,
$C_1^5 = \{3,6\}$,	$C_1^6 = \{3,5\}$.	$C_2^7 = \{4,5\}$.
$C_2^5 = \{2,4,5\}$.		

Pattern "1" of column 5 and pattern "1" of column 7 are designated dont care characters ("-") in the respective columns in Figure 7 (3). Next only C_1^6 is augmented with elements of the zeroset C_0^6 to obtain $\{1,2,3,4,5,6\}$.

$\mathcal{C}^5$	$\mathcal{C}^6$	$\mathcal{C}^7$
$C_0^5 = \{1\}$,	$C_1^{6'} = \{1,2,3,4,5,6\}$.	$C_2^7 = \{4,5\}$.
$C_2^5 = \{2,4,5\}$.		

Now, it can be easily verified that each pair of columns (5, 6, 7) is compatible and a compatible tree is shown in Figure 7 (3b).

1	2	3	4	5	6	7
1	3	1	1	0	0	1
1	3	1	1	2	0	1
2	2	3	2	1	1	1
2	1	2	0	2	0	2
2	1	2	0	2	1	2
2	2	3	2	1	0	1

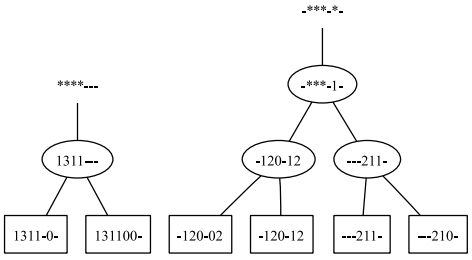
I

(1) The 6×7 character input matrix; 0 are the wild characters.

1	2	3	4	5	6	7
1	3	1	1	0	0	-
1	3	1	1	-	0	-
-	-	-	2	1	1	-
-	1	2	0	-	0	2
-	1	2	0	-	1	2
-	-	-	2	1	0	-

I^1

(2) $S = \{[1, 7]\}$.

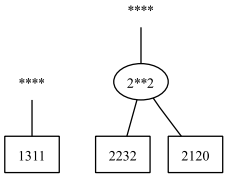


(2a) Forest on segment [1,7].

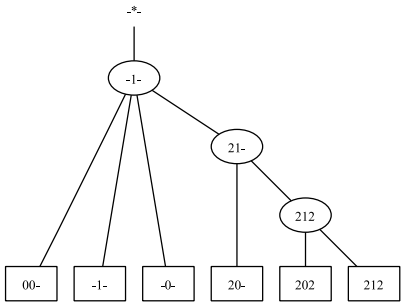
1	2	3	4	5	6	7
1	3	1	1	0	0	-
1	3	1	1	2	0	-
2	2	3	2	-	1	-
2	1	2	0	2	0	2
2	1	2	0	2	1	2
2	2	3	2	-	0	-

I^1 I^2

(3) $S = \{[1, 4], [5, 7]\}$.



(3a) Forest on segment [1,4].

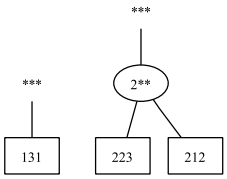


(3b) Tree on segment [5,7].

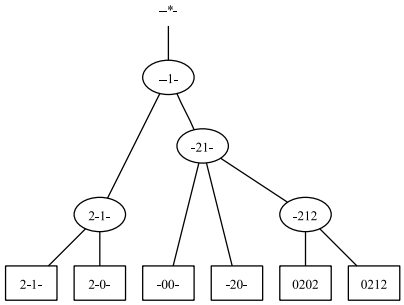
1	2	3	4	5	6	7
1	3	1	-	0	0	-
1	3	1	-	2	0	-
2	2	3	2	-	1	-
2	1	2	0	2	0	2
2	1	2	0	2	1	2
2	2	3	2	-	0	-

I^1 I^2

(4) $S = \{[1, 3], [4, 7]\}$.



(4a) Forest on segment [1,3].



(4b) Tree on segment [4,7].

FIG. 7. Compatible networks. Here we demonstrate the combinatorics involved in the network construction (in a more or less worst-case scenario). **(1)** The input matrix. **(2)** A possible designation of dont care characters to give a compatible network with no recombination (i.e., a single segment). Assuming at least one recombination in the data, **(3)** and **(4)** show two possible segmentations S_1 and S_2, with respective matrices I^1 and I^2. In all the networks, we omit the edge labels to avoid clutter.

5. (STEP 3) FORESTS TO NETWORKS: THE DSR ALGORITHM

Segmentation suggests a method to compute a compatible network: Given I, a possible segmentation and the corresponding forests (Fig. 2 (1)–(3)) are first constructed and then the compatible network (Fig. 1) is constructed from the forests. Thus we set the stage to solve the following problem.

Problem 4 (*Consensus Compatible Network Problem). Given two networks N_1 on a vertex set U and N_2 on a vertex set V, defined on the same set of samples, and with no common mutation edge labels, the task is to compute N_3 on some vertex set W, with two segments s_1 and s_2 such that for each edge label c_1 in N_1 and each edge label c_2 in N_2, the following holds:*

$$L(N_3, s_1, c_1) = L(N_1, s_1, c_1),$$

$$L(N_3, s_2, c_2) = L(N_2, s_2, c_2).$$

Overview of the approach. We solve this problem using a topology based method. Our approach is iterative, bottom-up working at one level of N_1 and N_2 at a time. The method gets its name from the need to give one of three possible "colors" (Dominant or Subdominant or Recombinant) assignment to nodes at each stage, which is central to this approach. Roughly speaking, a *dominant* node in W uses the edge labels of N_1 and N_2; a *subdominant* uses one of the edge labels of N_1 and N_2 (but not both); and a *recombinant* uses neither of the edge labels of N_1 and N_2 and is indeed a recombinant node in N_3. In the iterative procedure, the "color" of a dominant or a subdominant node may change to recombinant. We illustrate central ideas of the approach in Figure 8 with a concrete example (used earlier in Fig. 1).

In this example N_1 and N_2 are the networks (forests) shown in Figures 2 (1) and (2), respectively. For the DSR algorithm, the label of the leafnode is the set of samples or rows represented by that node. We begin by considering the bottommost level in both N_1 and N_2 and computing the intersection matrix X. Let P_u be the leaf nodes in N_1 and P_v be lefnodes in N_2. For convenience, the P's of iteration i are written as P_u^i and P_v^i. P_u^i and P_v^i are updated through the iterations as follows:

$$P_u^i = \{u \in U \mid \text{for all descendants } x \text{ of } u \text{ in } N_1,\ x \in P_u^{i'} \text{ holds for some } i' < i\},$$

$$P_v^i = \{v \in V \mid \text{for all descendants } x \text{ of } v \text{ in } N_2,\ x \in P_v^{i'} \text{ holds for some } i' < i\}.$$

Compatible DSR assignment of X. An appropriate assignment of DSR is required so that a network N_3 can actually be built. The DSR assignment of matrix X is compatible if it satisfies the following two conditions:

1. Each row and each column in matrix X has at most one dominant. If there is no dominant, then it has at most one subdominant.
2. A non-recombinant element can have another non-recombinant in its row or its column but not both.

It can be verified that if any one of the above conditions is violated, it would be impossible to construct a compatible network N_3.

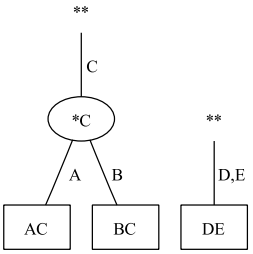
In Figure 8, the dominant entries are shown enclosed in a box. We assign a label to the entries of matrix X that is used in the next level. Note that a recombinant need not be labeled since it is not used anymore in the subsequent levels. Thus, {2} is not labeled in Figure 8 (5a), and t is a subdominant; p, q, and r are dominants. To avoid clutter, we skip the details and appeal to the reader's intuition. Note that there can be no algorithm that guarantees an optimal solution and we adopt heuristics that work well in most cases.

Although in the discussion above N_1 and N_2 are forests, the same method can be extended to the case when one or both are not forests. The detail is tedious and we skip the discussion in this paper.

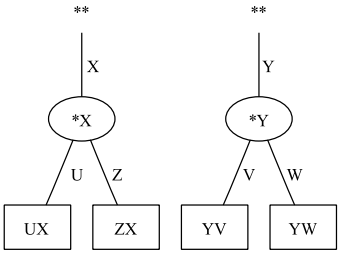
The correctness of the DSR algorithm follows from the two observations which can be verified.

Observation 6 (*Incomparable $\boldsymbol{P}$'s). If $u_1, u_2 \in P_u^i$, at some iteration i, then u_1 and u_2 are incomparable in N_1. Similarly for $v_1, v_2 \in P_v^i$.*

Observation 7 (*Incomparable $\boldsymbol{w}$'s). If two solid characters c_1 and c_2 are edge labels at position j, incident on nodes w_1 and w_2 in N_3, then w_1 and w_2 are incomparable in N_3.*



(1) N_1.



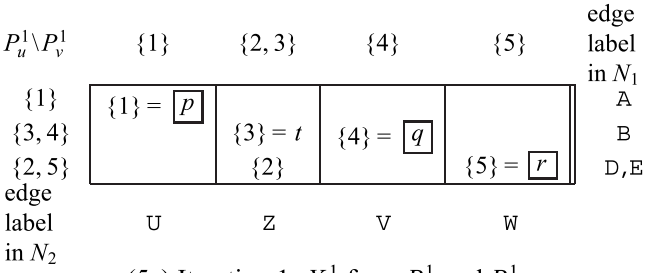
(2) N_2.

leaf node label	P_1^1
AC	$\{1\}$
BC	$\{3,4\}$
DE	$\{2,5\}$

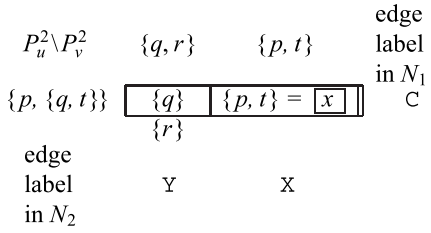
(3) Leafnodes to sets in N_1.

leaf node label	P_2^1
UX	$\{1\}$
ZX	$\{2,3\}$
YV	$\{4\}$
YW	$\{5\}$

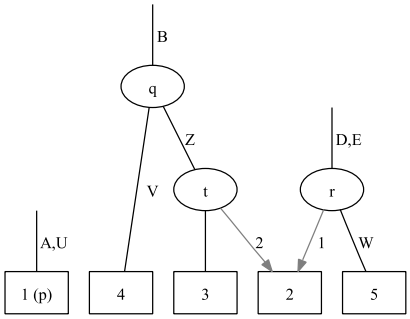
(4) Leafnodes to sets in N_2.



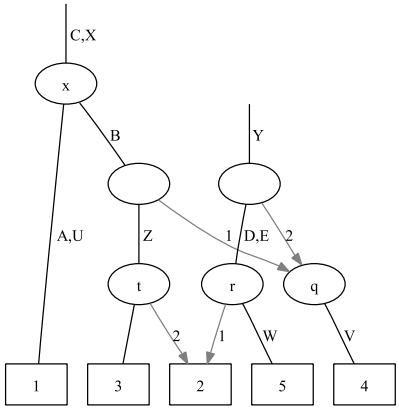
(5a) Iteration 1: X^1 from P_u^1 and P_v^1.



(6a) Iteration 2: X^2 from P_u^2 and P_v^2.



(5b) Topology corresponding to (5a).



(6b) Topology using (5b) and (6a).

FIG. 8. Workings of the DSR Algorithm. N_1 and N_2 with the input samples of Figure 1. Iteration **(1)** is shown in **(5)** and the final iteration **(2)** is shown in **(6)**. The compatible network N_3 is shown in 6b, at the end of two iterations of the algorithm.

6. EXPERIMENTAL RESULTS

Since it is difficult to accurately assess a model directly on real data sets, we evaluate the performance of our model on simulated populations that display expected human genome sequence variations: we measure the results of our model against the known characteristics of these simulated data sets.

6.1. Population simulations

Assessing the performance of the model would imply working with sets of sequences where the exact location of recombination events is known. Since such information is not available for naturally occurring human sequences, we have resorted to computer simulations. Coalescence theory has provided a fast, efficient framework for the simulation of extant human variation, by specifying the distribution of the lengths of the coalescent tree of a population sample. In a coalescent model with recombination, coalescent and recombination events are placed at random with predetermined probabilities, thus generating an ARG rather than a pure tree. On top of the ARG, mutational random events can also be added at preset rates. The shape of the network itself is modulated by demography, and relatively complex demographic histories can be simulated easily.

We have used the COSI program (Schaffner et al., 2005) to generate problem sequence sets with known recombination events. We used a demographic model with three phases: two stationary phases separated by a bottleneck event. In the first stationary phase, diversity is generated by mutation and recombination; the bottleneck has the effect of subsampling from that diversity and seeding the last phase in which mostly recombination will act, since we filter by SNP allele frequency, thus rejecting the more recently created polymorphisms. The time scale, effective population size, and mutation and recombination rates are roughly based on current estimates, although recombination rate is clearly below the average genome rate, which is driven by the accumulation of events in the so-called recombination hotspots. Since recombination is probably saturated at hotspots, they are not particularly informative for phylogeographic reconstructions and they do not represent the conditions in which the method is meant to be applied. The mutation rate (μ) was set to 1×10^8 mutations per nucleotide site per generation. The recombination rate was set to 1×10^9 recombinations per nucleotide site per generation. The mutation generation phase has an effective population size $N_e = 10{,}000$. The bottleneck occurred 3,000 generations ago and shrinks to effective population of 100 during 100 generations. Then, 2,900 generations ago, the effective populations recovers to 10,000 up to the present. Needless to mention, simulations can be done in more sophisticated settings; nonetheless our simulations give datasets similar to the haplotypes observed in Hapmap.

6.2. Performance measure

The central problem in this paper was motivated by the need for an analysis of haplotype data in terms of recombinations. The two primary aspects were to identify (1) the location of recombination breakpoints and (2) to distinguish the *ancestor* and the *derivative*.

Using our parlance, the derivative sequence is a recombinant and the ancestor corresponds to an ancestral node in the ARG network. Figure 9 shows an example describing the terms and the issues involved for

(1)	ACGGTGGTTTGGGTTGT	(1)	ACGGTGGTTTGGGTTGT	(1)	ACGGTGGTT|TGGGTTGT
(2)	TTGGCCAGGTTACTGTT	(2)	TTGGCCAGGTTACTGTT	(2)	TTGGCCAGG|TTACTGTT
(3)	ACGGTGGTTTTACTGTT	(3)	ACGGTGGTT|TTACTGTT	(3)	ACGGTGGTTTTACTGTT
			Derivatives: (3), Ancestors: (1) and (2).		Derivatives: (1) and (2), Ancestors: (3).
	(a)		(b)		(c)

FIG. 9. Ancestor-Derivative Dilemma. **(a)** Three input haplotypes. Two possible scenarios for ancestor and derivative designations shown in (b) and (c). The point of recombination is shown by the symbol "|" in both scenarios. **(b)** Here haplotypes (1) and (2) are considered ancestral with (3) as a derivative (or recombinant) of (1) and (2). **(c)** Only haplotype (3) is ancestral, and both (1) and (2) are derivatives of (3) and a haplotype that is not included in this sample space of 3.

distinguishing the two. This example shows three input haplotypes and two possible designations of ancestor and derivative. In the absence of additional information, it is difficult to choose between the two possibilities in the example. Our contention is that the estimation of the ARG phylogenetic network resolves this issue, almost completely. In all the simulated data we observed that, in all the true-positives, the ancestor and the derivative (recombinant) are identified correctly. So in the rest of the section we focus on the identification of breakpoint locations.

We show some sample compatible networks estimated by our model in three data sets on small fragments of the input in Figure 10. Such networks are analyzed by hand for the analysis discussed below.

Quantitative Analysis. In this section we analyze our results on simulated data in a model-independent manner (for a model-dependent analysis, see Appendix). Given a haplotype of length m, there are potentially $m-1$ recombination breakpoints. We compare the model's predicted recombination breakpoints with that is known from the simulation from this universe of $m-1$ breakpoints. In the next paragraph, we discuss a way of meaningfully reducing this universe size using *switch of patterns*.

Mutations such as $\texttt{A} \rightarrow \texttt{C} \rightarrow \texttt{A}$, is usually impossible to detect (unless there is other compelling accompanying evidence). Also $\texttt{A} \rightarrow \texttt{C} \rightarrow \texttt{G}$, is seen as one mutation and not two. To avoid similar confusions, most times an *infinite sites model* is assumed for mutations in literature (Felsenstein, 2004): this states that each *visible* mutation occurs exactly once. Similarly, such recurrent recombinations are also impossible to detect. For instance, if both the ancestors of a recombinant are identical, then the recombination, not surprisingly, is not detectable by most models. We call such undetectable recombinations *invisible*.

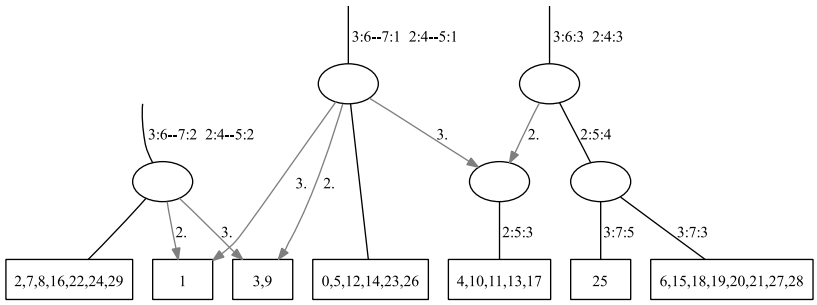
Here we follow a population geneticist's intuition in focussing on the potentially visible ones: A visible recombination is characterized by a *switch of pattern* along the haplotype. Note that a single *pattern* appears in multiple haplotypes (or haplotype clusters). But not all such switches represent recombination events. Also, all the instances of the pattern may not be identical (but are similar and belong to a haplotype-cluster). We make the tacit assumption that all visible recombinations must necessarily be covered by these breakpoints. However, it is quite possible that multiple recombinations may occur at the same point.

In the following analysis, the location of a pattern switch is treated as the universe of all possible recombination points (instead of $m-1$). We compare the compatible phylogenetic network (model) with the simulation ARG (gold standard) focussing on the recombinations. This is a painstaking comparison done by hand to confirm the ancestor-derivative designation for all the true-positives. Since, we have not found any error in these designations, we do not explicitly report these in our analysis and simply give the count of the recombinations. We summarize the results for some randomly chosen simulation datasets in Figure 11. Each data set is generated by COSI using parameters discussed in Section 6.1, to reflect the variations in human genome sequences.

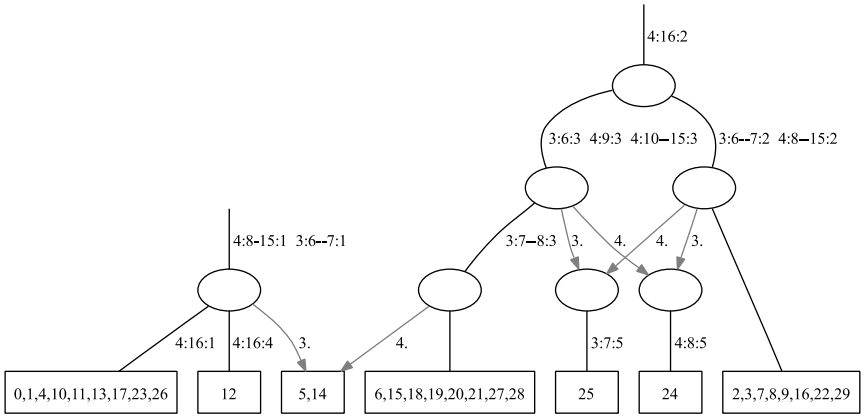
U is the number of switch of patterns observed in the given data. Amongst the model's reported breakpoints, *true positives* (TP) are the correct predictions and *false positives* (FP) are the wrong predictions. The breakpoints that the model misses are reported as *false negatives* (FN) and all the remaining points ($U-$ TP $+$ FP $+$ FN) are the *true negatives* (TN). To summarize, the four values can be viewed as:

	Simulation (gold standard)	
	Positive	*Negative*
Model estimation		
Positive	*True positives* (TP)	*False positives* (FP)
Negative	*False negatives* (FN)	*True negatives* (TN)

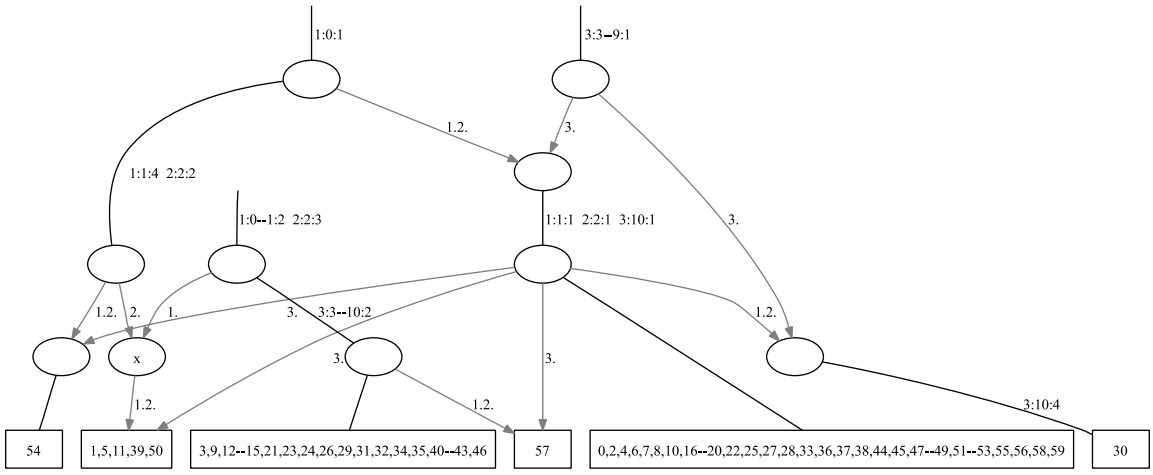
Sensitivity is computed as TP/(TP + FN) and specificity is computed as TN/(TN + FP). Figure 11 shows the results on ten data sets. We observe an average sensitivity of 0.15% and an average specificity of 88.95% in the data.



(1) Compatible network with two segments (labels 2, 3) on 30 haplotype clusters and 20 SNPs.



(2) Compatible network with two segments (labels 3, 4) on 30 haplotype clusters and 55 SNPs.



(3) Compatible network with three segments (labels 1, 2, 3) on 60 haplotype clusters and 50 SNPs.

FIG. 10. Three sample networks on short fragments in three data sets **(1–3)**. An element in the mutation edge label is interpreted as $s{:}j{:}val$, i.e., segment s and column j has a value val. The labels of the leaf node (rectangular boxes) are the haplotype cluster numbers. Also $x{-}y$ in the edge or node labels is to be interpreted as $x, x+1, \ldots, y-1, y$. Also, the node marked x in (3) is an example of an internal node where the values of the position in segment 3 is immaterial since it is not transmitted to any of the extant sequences in the leaf nodes.

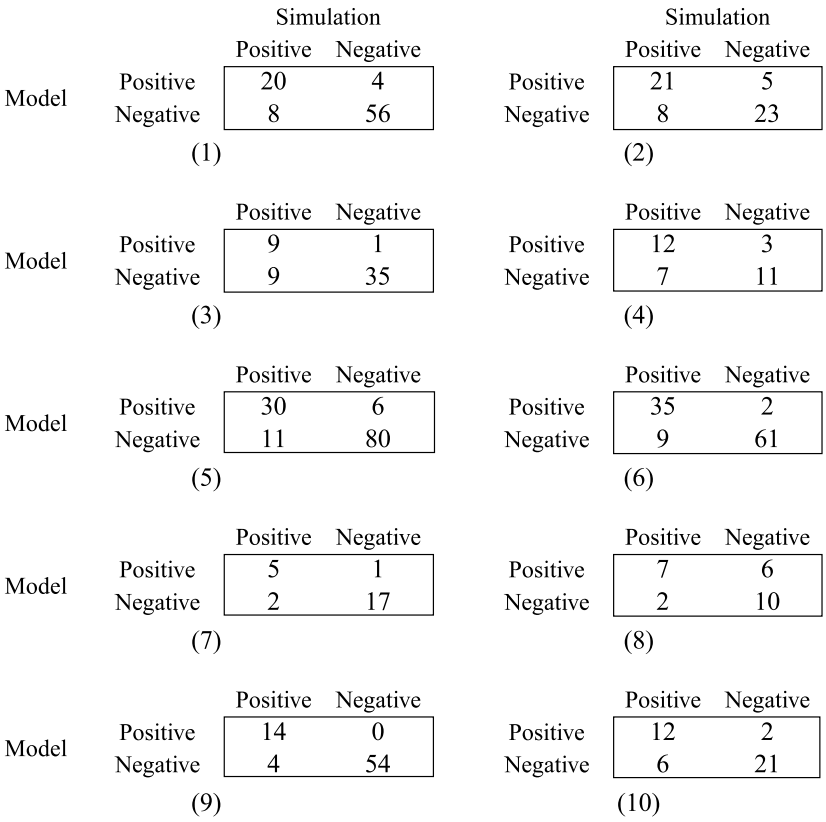


FIG. 11. The contingency tables for 10 data simulation data sets each with the parameters discussed in Section 6.1 **(1–10)**. In each, the samples have no less than 25 and no more than 100 distinct haplotypes and an average of 200 SNP's. In each, a grain size $g = 5$ is used.

7. CONCLUSION

Recombination, by itself, is not a phenomenon that can be easily recognized as a product status in a sequence. Only the analysis of multiple sequences may be of help in recognizing points where patterns change, in some of them, as a complex function of some of the remaining sequences. Thus detection of recombination becomes a statistical or combinatorial (not a biological) challenge. The challenge is to identify by statistical or combinatorial means points in the data that are related to the genetic input of recombinations.

We have presented a new method for detecting recombination events that combines SNP patterns with phylogeny: it is appealing in its simplicity of the underlying algorithms. Through our experimentations, we observed that the method is fairly robust in the sense that it is insensitive to most parameter changes. For instance, we observed that for the simulations we experimented with, a grain size $g = 5$ could be uniformly used in all the cases.

In summary, our method provides a promising avenue for the detection of recombination and its future incorporation in phylogeography. We hope to rescue recombination from its actual pariah status in phylogeography, and pave the way for the full use of multiple independent autosomal loci in reconstructing population history.

8. APPENDIX

In the main body of the paper, we evaluate our models by focusing primarily on recombinations. However, the reality is that the model estimates other characteristics as well, such as clusters, the phylogenetic

structure, and so on, and the final numbers on recombinations is not independent of these intermediate results of the multi-stage approach. An analysis of each stage provides invaluable insights into the strengths and weaknesses of our model. So, here, we focus on these intermediate by-products.

We present the results for a simulated data set that has 160 SNPs and 20 haplotypes. The ARG has six segments (with non-recombining history) in the simulation as follows:

$$[0-83], \quad [84-86], \quad [87-101], \quad [102-135], \quad [136-151], \quad [152-159].$$

8.1. Haplotypes to character matrix

Here we focus on the clustering of the haplotypes and compare it with hand curated haplotype clusters. Recall that clusters are groups of haplotypes that possibly do not have any recombinations in their common history and can be, loosely speaking, painted with the same color in a mosaic model. The first column (grain values 1, 2, and 3) gives exactly the same clusters as is produced by hand. As the grain is made coarser with values 4 and 5, overlapping clusters (with alternative hypotheses) are obtained and these are marked with an asterisk below.

Grain $= 1, 2, 3$	*Grain* $= 4$	*Grain* $= 5$	*Grain* $= 6$
18	18	*0, 6, 18	
13	13		
0, 6	0, 6	*0, 6, 13	0, 6, 13, 18
5, 12	*5, 12	*5, 12	5, 7, 12
7	*5, 7	*5, 7	
4, 9, 12, 10, 17	4, 9, 12, 10, 17	4, 9, 12, 10, 17	4, 9, 12, 10, 17
15, 16	15, 16	15, 16	15, 16
8	8	8	8
3	3	3	3
11, 19	11, 19	11, 19	11, 19
14	14	14	14
11	11	10	8

8.2. Segmentation

Here we give the segmentation obtained for different grain sizes.

Grain	*Segments (approximate)*						
	[0-83]	[84 − 86]	[87 − 101]	[102 − 135]	[136 − 151]	[152 − 159]	
1	← − →	← − →	← − →	← − →	← − →	← − →	6
2	← − →	← − − − − − − →		← − − − − − − →		← − →	4
3	← − →	← − − − − − − →		← − − − − − − − − − − →			3
4	← − − − − − − − − − − − →			← − − − − − − − − − − − →			2
5	↔	↔	← − − − − − − →		← − − − − − − − − − − − →		4
6	← − − − − − − − − − − − →			← − − − − − − − − − − − →			2

g is the grain size and the number of segments for the grain size g is shown ar the right most column. For example, grain $g = 2$ gives 4 segments in the segmentation as:

$$[0, 83]; \quad [84, 101]; \quad [102, 151]; \quad [152, 159].$$

In this data set, note that only for grain size 5 is a false positive segmentation is introduced (at the very first segment). Otherwise, every other error is a false negative error. Hence from this we conclude that the chances of missing a correct segmentation is more likely than introducing a wrong one.

8.3. Quantitative assessment of (segmented) ARG: a weighted Robinson Foulds measure

We use a weighted version of the Robinson-Foulds tree distance: the details of this weight computation is discussed here. The use of segmentation in our model facilitates a much more straightforward computation than a tripartition-based distance measure (Moret et al., 2004).

Given two sets g and h, at least one of which is non-empty, the classical Jaccard's index of similarity between the two is defined as:

$$J(g,h) = \frac{|g \cap h|}{|g \cup h|}. \tag{1}$$

Recall that $0 \leq \mathcal{J}(g,h) \leq 1$, with a value of 1 as a perfect match, i.e., $g = h$ and a value of 0 as a perfect mismatch, i.e. $g \cap h = \emptyset$. We use this Jaccard index to assign weight to the Robinson-Foulds tree distance (Robinson and Foulds, 1981). In the unweighted version,

$$\mathcal{J}(g,h) = \begin{cases} 1 & \text{if } J(g,h) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

The weighted version, $\mathcal{J}(g,h) = J(g,h)$, is a more accurate similarity measure and is used in this analysis.

We use the following measure to compare two collection G and H of possibly overlapping sets of clusters (of Section 8.1). A mapping $F(G) \rightarrow H$ is defined as

$$F(g) = \text{argmax}\left(\max_{h \in H} \mathcal{J}(g,h)\right). \tag{2}$$

The distance of H from G using Equations (1) and (2) is:

$$dis(G,H) = \frac{\sum_{g \in G} \mathcal{J}(g, F(g))}{|G|}.$$

We take the average of $dis(G,H)$ and $dis(H,G)$ so that $dis(G,H) = dis(H,G)$. The mapping $F'(H) \rightarrow G$ is defined similarly. The score for the pair G and H, written as $score(G,H)$ is:

$$score(G,H) = \frac{dis(G,H) + dis(H,G)}{2} = \frac{\sum_{g \in G} \mathcal{J}(g, F(g)) + \sum_{h \in H} \mathcal{J}(h, F'(h))}{|G| + |H|}.$$

Each segment has a weight w_i which is equal to the number of SNPs in the segment and let G_i and H_i be the collection of clusters for the segment in the estimated and true model respectively. The similarity score ***RF*** with the known ARG (from the simulation) is defined as:

$$\boldsymbol{RF} = \frac{\sum_{(seg\ i)} score(G_i, H_i) w_i}{\sum_{(seg\ i)} w_i}.$$

Grain	*Score*(G_i, H_i) for each *seg i*							***RF***
size	84		3	15	34	16	8	←seg wt (w_i)
1	1.0		1.0	0.88	1.0	0.98	1.0	**0.99**
2	1.0		0.74; 0.87		0.68; 0.82		0.70	**0.88**
3	1.0		0.65; 0.93		0.77;0.67;0.66			**0.89**
4	1.0;0.49;0.70				0.61;0.68;0.63			**0.83**
5	0.94 (55)	0.83 (29)	0.59;0.88		0.65;0.68;0.55			**0.80**
6	0.98;0.61;0.83				0.66;0.60;0.51			**0.83**

ACKNOWLEDGMENTS

Part of this work was done at IBM TJ Watson Research Center during an intership by M.M. The study was supported by the Spanish Ministry of Science (Ph.D. fellowship FPU AP2006-03268) for the second author (M.M.) and by the Spanish Ministry of Science (grant BFU-63657) for the third author (F.C.). We are also thankful to Ajay Royyuru for his insightful comments on the work and bringing our attention to the tool *Graphviz* (Gansner et al., 1993): most of the networks in this paper are drawn using this software.

DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Behar, D.M., Rosset, S., Blue-Smith, J., et al. 2007. The genographic project public participation mitochondrial DNA database. *PLoS Genet.* 3.

Daly, M.J., Rioux, J.D., Schaffner, S.F., et al. 2001. High-resolution haplotype structure in the human genome. *Nat. Genet.* 29, 229–232.

Felsenstein, J. 2004. *Inferring Phylogenies*. Sinauer Associates, Sutherland, MA.

Gabriel, S.B., Schaffner, S.F., Nguyen, H., et al. 2002. The structure of haplotype blocks in the human genome. *Science* 296, 2225–2229.

Gansner, E.R., Koutsofios, E., North, S.C., et al. 1993. A technique for drawing directed graphs. *Software Eng.* 19, 214–230.

Gusfield, D., Bansal, V., Bafna, V., et al. 2007. A decomposition theory for phylogenetic networks and incompatible characters. *J. Comput. Biol.* 14, 1247–1272.

International HapMap Consortium. 2007. A second generation human haplotype map of over 3.1 million SNPs. *Nature* 449, 851–861.

Mannila, H., Koivisto, M., Perola, M., et al. 2003. Minimum description length block finder, a method to identify haplotype blocks and to compare the strength of block boundaries. *Am. J. Hum. Genet.* 73, 86–94.

Minichiello, M.J., and Durbin, R. 2006. Mapping trait loci by use of inferred ancestral recombination graphs. *Am. J. Hum. Genet.* 79, 910–922.

Moret, B.M.E., Nakhleh, L., Warnow, T., et al. 2004. Phylogenetic networks: modeling, reconstructibility, and accuracy. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 1, 13–23.

Patil, N., Berno, A.J., Hinds, D.A., et al. 2001. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science* 294, 1719–1723.

Robinson, D.F., and Foulds, L.R. 1981. Comparison of phylogenetic trees. *Math. Biosci.* 53, 131–147.

Roubinet, F., Despiau, S., Calafell, F., et al. 2004. Evolution of the O alleles of the human ABO blood group gene. *Transfusion* 44, 707–715.

Schaffner, S.F., Foo, C., Gabriel, S., et al. 2005. Calibrating a coalescent simulation of human genome sequence variation. *Genome Res.* 15, 1576–1583.

Song, Y.S., Lyngso, R., and Hein, J. 2006. Counting all possible ancestral configurations of sample sequences in population genetics. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 3, 239–251.

Ukkonen, E. 2002. Finding founder sequences from a set of recombinants. *Proc. WABI* 2542, 277–286.

Wang, N., Akey, J.M., Zhang, K., et al. 2002. Distribution of recombination crossovers and the origin of haplotype blocks: the interplay of population history, recombination, and mutation. *Am. J. Hum. Genet.* 71, 1227–1234.

Watterson, G.A., and Guess, H.A. 1977. Is the most frequent allele the oldest? *Theor. Popul. Biol.* 11, 141–160.

Wilson, E.O. 1965. A consistency test for phylogenies based on contemporaneous species. *Syst. Zool.* 14, 214–220.

Wu, Y., and Gusfield, D. 2007. Improved algorithms for inferring the minimum mosaic of a set of recombinants. *Lect. Notes Comput. Sci.* 4580, 150–161.

Zhang, K., Deng, M., Chen, T., et al. 2002. A dynamic programming algorithm for haplotype block partitioning. *Proc. Natl. Acad. Sci.* 99, 7335–7339.

Address reprint requests to:
Dr. Laxmi Parida
Computational Biology Center, IBM TJ Watson Research
Route 134
Yorktown Heights, NY 10598

E-mail: parida@us.ibm.com