INFORMS Journal on Computing
Vol. 16, No. 4, Fall 2004, pp. 459–469
ISSN 0899-1499 | EISSN 1526-5528 | 04 | 1604 | 0459

DOI 10.1287/ijoc.1040.0099

The Fine Structure of Galls in Phylogenetic Networks

Dan Gusfield, Satish Eddhu
Department of Computer Science, University of California, Davis, Davis, California 95616, USA
{gusfield@cs.ucdavis.edu, eddhu@cs.ucdavis.edu}

Charles Langley
Division of Evolution and Ecology, University of California, Davis, Davis, California 95616, USA,
chlangley@ucdavis.edu

A phylogenetic network is a generalization of a phylogenetic tree, allowing properties that are not tree-like. With the growth of genomic data, much of which does not fit ideal tree models, there is greater need to understand the algorithmics and combinatorics of phylogenetic networks (Posada and Crandall 2001, Schierup and Hein 2000).

Wang et al. (2001) studied the problem of constructing a phylogenetic network for a set of n binary sequences derived from the all-zero ancestral sequence, when each site in the sequence can mutate from zero to one at most once in the network, and recombination between sequences is allowed. They showed that the problem of minimizing the number of recombinations in such networks is NP-hard, but introduced a special case of the problem, i.e., to determine whether the sequences could be derived on a phylogenetic network where the recombination cycles are node-disjoint. Wang et al. (2001) provide a sufficient, but not a necessary test, for such solutions. Gusfield et al. (2003, 2004) gave a polynomial-time algorithm that is both a necessary and sufficient test. In this paper, we study in much more detail the fine combinatorial structure of node-disjoint cycles in phylogenetic networks, both for purposes of insight into phylogenetic networks and to speed up parts of the previous algorithm. We explicitly characterize all the ways in which mutations can be arranged on a disjoint cycle, and prove a strong necessary condition for a set of mutations to be on a disjoint cycle.

The main contribution here is to show how structure in the phylogenetic network is reflected in the structure of an efficiently-computable graph, called the *conflict graph*. The success of this approach suggests that additional insight into the structure of phylogenetic networks can be obtained by exploring structural properties of the conflict graph.

Key words: molecular evolution; phylogenetic networks; ancestral recombination graph; recombination; SNP; bi-convex graph
History: Accepted by Harvey J. Greenberg, Guest Editor; received August 2003; accepted November 2003.

1. Introduction and Main Results

With the growth of genomic data, much of which does not fit ideal evolutionary-tree models, and the increasing appreciation of the genomic role of such phenomena as recombination, recurrent and back mutation, horizontal gene transfer, gene conversion, and mobile genetic elements, there is greater need to understand the algorithmics and combinatorics of phylogenetic networks (Posada and Crandall 2001, Schierup and Hein 2000). Recombination is particularly important because it is the key element needed for techniques that are widely hoped to locate genes influencing genetic diseases. The key to locating these genes is to understand and use the patterns of recombination in the genetic *experiments* done by nature and history. However, very little is known about the combinatorial structure of phylogenetic networks. A seminal paper (Wang et al. 2001) began a focus on phylogenetic networks in the case that the underlying cycles are node-disjoint, and they introduced the problem of determining if a set of sequences could be derived on such a phylogenetic network. The algorithm given provides a sufficient but not a necessary test for that problem. A faster, complete (both necessary and sufficient) algorithm for that problem, is given in Gusfield et al. (2003, 2004). Other papers related to phylogenetic networks include Hein (1990, 1993), Song and Hein (2003), Myers and Griffiths (2003), Kececioglu and Gusfield (1998).

1.1. Formal Definition of a Phylogenetic Network

There are four components needed to specify a phylogenetic network: A directed acyclic graph (no directed cycles, but the underlying undirected graph can have cycles); an assignment of mutations or sites (integers) to edges; an assignment of a sequence to each

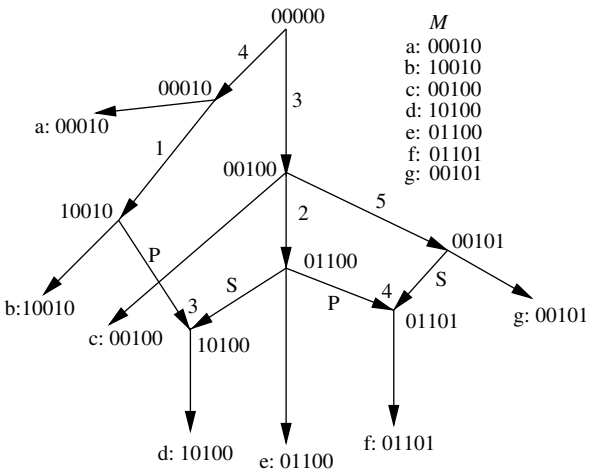

Figure 1 A Phylogenetic Network *N* with Two Recombination Nodes
Note. The matrix of sequences *M* that are derived by *N* is shown at the right.

nonrecombination node; and an assignment of a recombination point and a sequence to each recombination node. We will define each of these components in turn. See Figure 1 for an example of a phylogenetic network.

An (n, m)-phylogenetic network N is built on a directed acyclic graph containing exactly one node (the root) with no incoming edges, a set of internal nodes that have both incoming and outgoing edges, and exactly n nodes (the leaves) with no outgoing edges. Each node other than the root has either one or two incoming edges. A node x with two incoming edges is called a *recombination* node.

Each integer (site) from 1 to m is assigned to exactly one edge in N, but for simplicity of exposition, none are assigned to any edge entering a recombination node. There may be additional edges that are assigned no integers. We use the terms *column* and *site* interchangeably.

Each node in N is labeled by an m-length binary sequence, starting with the root node, which is labeled with the all-zero sequence. Because N is acyclic, the nodes in N can be topologically sorted into a list, where every node occurs in the list only after its parent(s). Using that list, we can constructively define the sequences that label the nonroot nodes, in order of their appearance in the list, as follows:

(a) For a nonrecombination node v, let e be the single edge coming into v. The sequence labeling v is obtained from the sequence labeling v's parent by changing from zero to one the value at position i, for every integer i assigned to edge e. This corresponds to a mutation at site i occurring on edge e.

(b) Each recombination node x is associated with an integer r_x (denoted r when x is clear by context) between two and m inclusive, called the *recombination point* for x. For the recombination at node x, one of the two sequences labeling the parents of x must be designated P and the other designated S. Then the sequence labeling x consists of the first $r_x - 1$ characters of P, followed by the last $m - r_x + 1$ characters of S. Hence P contributes a *prefix* and S contributes a *suffix* to x's sequence. The resulting sequence that labels x is called a *recombinant sequence*.

Recombinations occur in the phylogenetic network at a recombination *node* and this distinguishes a change of state due to recombination from a change of state due to mutation.

The sequences labeling the leaves of N are the extant sequences, i.e., the sequences that can be observed. Note the assumption that the ancestral sequence (at the root node) is the all-zero sequence.

Note that in Figure 1, the node with sequence label 01100 is sequence S for the left recombination node, and is sequence P for the right recombination node. The recombination points are three and four for the left and right recombination nodes respectively, and are written just above the recombination nodes. In this example, every label of an interior node also labels a leaf, but that is not a general property of phylogenetic networks.

Definition 1.1. An (n, m)-phylogenetic network N *derives (or explains)* a set of n sequences M if and only if each sequence in M labels exactly one of the leaves of N.

With these definitions, a classic perfect phylogeny is a phylogenetic network that is topologically a directed, rooted tree, i.e., lacking any cycles in the underlying (undirected) graph. What we have defined here as a phylogenetic network is the digraph part of the stochastic process called an *ancestral recombination graph* in the population-genetics literature (see Norborg and Tavare 2002, for example).

The biological interpretation of a phylogenetic network N that derives M is that N is a possible history of the evolution of the sequences in M, under the assumptions that there is a single, known ancestral sequence (assumed to be all-zero for convenience); that for any site in the sequences there is exactly one point in the history (recorded on an edge) where that state of that site mutates (due to a point mutation) from 0 to 1; and that two sequences are permitted to recombine (during meiosis) in a single equal-crossover event. Each site in the sequence represents a SNP (single nucleotide polymorphism), i.e., a site where two of the four possible nucleotides appear in the population with a frequency above some fixed threshold.

These biological interpretations are most realistic when the sequences come from individuals in the same population, and the individuals have a common ancestor in the relatively recent past. We should note that meiotic recombination is a phenomenon that

occurs inside a single species, while the term *phylogeny* most correctly refers to evolutionary history involving several species. Therefore, it is not completely correct to use the term *phylogenetic network* for a history of meiotic recombinations. However, in computer-science literature (and in some parts of biology), the term *phylogeny* has come to be synonymous with *evolutionary tree* or *evolutionary history*, regardless of the source of the data being studied. Similarly, the term *phylogenetic network* has been introduced in earlier papers to refer to evolutionary trees with the incorporation of recombination. We continue the abuse of the term *phylogenetic* in this paper.

The strongest assumption above is that the ancestral sequence is known for the network. It is often possible to determine the ancestral sequence, and the root of the network, through the use of an *outgroup*, i.e., an individual who relates to the individuals under study, through the root node of the network.

Interest in phylogenetic networks comes partly from a desire to reconstruct the evolutionary history of a set of molecular sequences under a model that is more complete than is the perfect phylogeny (tree) model. But there are also more applied uses of phylogenetic networks. For example, in a population of *unrelated* individuals, we want to determine which parts of the individuals' genomes came from a common ancestor. This determination helps locate regions in the genome associated with genes contributing to an observable trait (for example, a disease). Recombination in the population is a key element making this possible, and understanding the history of the recombinations is the key to doing this kind of mapping.

1.2. Which Phylogenetic Networks Are Biologically Informative?

It is easy to show that, for every binary matrix M, there is a phylogenetic network N that derives M using $\Theta(nm)$ recombination nodes, but that is not of great interest because in most evolutionary histories the number of recombinations is thought to be relatively small (on the order of the number of mutations). Hence a more biologically informative problem is to find, for input M, a phylogenetic network that generates M, and that either has some biologically-motivated structure, or uses the *minimum* number of recombinations.

1.2.1. Galls in Phylogenetic Networks. If M cannot be derived on a perfect phylogeny (a phylogenetic network with no recombinations), some cycles in the underlying graph will be needed, but we would like to deviate from a tree by as little as is necessary. Rather than having a network with a complex interleaving of cycles, it is preferable (if possible) to have a tree with some extra edges, each creating a disjoint cycle. Here we define this more formally.

Definition 1.2. In a phylogenetic network N, let w be a node that has two paths out of it that meet at a recombination node x. Those two paths together define a "recombination cycle" Q. Node w is called the "coalescent node" of Q, and x is the recombination node of Q. The two sides (the P side and the S side) of the cycle can be identified by the node labeled with sequence P, and the node labeled with sequence S.

Clearly, a phylogenetic network must contain some recombination cycle(s) if the input sequences M cannot be derived on a perfect phylogeny.

Definition 1.3. A recombination cycle in a phylogenetic network that shares no nodes with any other recombination cycle is called a "gall" (imagine a wasp's gall in a tree). We say a site i "appears" on a gall Q if i labels one of the edges of Q. We use the term "recombination cycle" for general phylogenetic networks.

A gall in a phylogenetic network is locally a small deviation from a perfect phylogeny—the subnetwork that has a recombination cycle, but not a *tangle* of multiple recombination cycles.

Definition 1.4. A phylogenetic network is called a "galled tree" if every recombination cycle is a gall. See Figure 2.

1.3. Prior Algorithm

In Gusfield et al. (2003) we developed an efficient algorithm ($O(nm + n^3)$-time) that determines whether or not a set of input sequences M can be derived on a galled tree. We have more recently shown (Gusfield et al. 2004) that when there is a galled tree for input M, the algorithm creates a galled tree that uses the minimum possible number of recombinations, over

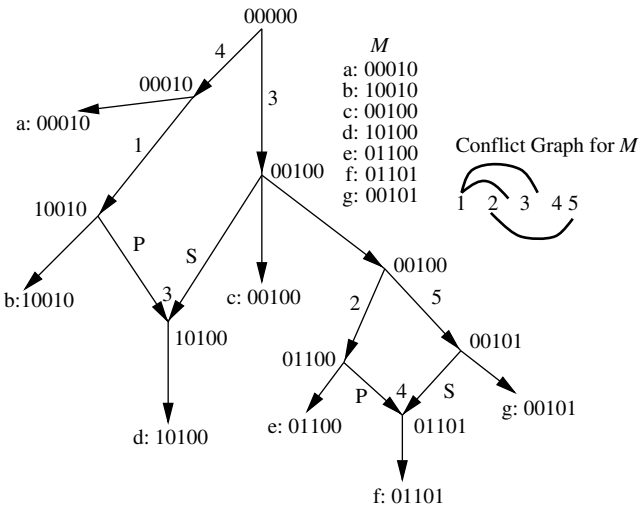

Figure 2 A Galled-Tree Deriving the Same Sequences as the Phylogenetic Network in Figure 1

Note. Unlike the example shown here, in general the recombinant sequence exiting a gall may be on a path that reaches another gall.

all possible phylogenetic networks for M with the all-zero ancestral sequence. In Gusfield (2004), we extended these results to the case that the ancestral sequence is not known in advance.

More generally, we showed that if a set of sites can be arranged on a gall in some phylogenetic network, then those sites must correspond to all the nodes in a single connected component of a *conflict graph*, and that corresponding connected component must have certain structural properties, such as being bipartite. We also gave an algorithm to determine how those sites can be arranged on a gall, if possible. Hence galls are a structural feature (and essentially the only one) of phylogenetic networks that are fairly well understood.

1.4. Main Results

Because so little about the structure of phylogenetic networks is understood, and because phylogenetic networks are important for a variety of biological applications, we want to continue developing our understanding of galls in general phylogenetic networks, and their relation to the conflict graph.

In this paper, through a deeper analysis of the combinatorial structure of galls and conflict graphs, we strengthen the previous necessary conditions for a set of sites to be arrangeable on a gall in some phylogenetic network, and we develop a faster and more informative algorithm for arranging the sites on a gall. In particular, the algorithm in this paper for arranging c sites on a gall takes $O(cn + c\log c)$ time, compared to the prior algorithm which takes $O(c^2n)$ time. Further, we prove that a connected component of the conflict graph that corresponds to a gall in some phylogenetic network must be *bi-convex*. It also follows that M can be derived on a galled tree only if the conflict graph for M is bi-convex. That result also has consequences for speeding up the solution to the maximum site-consistency problem (Day and Sankoff 1986).

2. Combinatorial Background

We organize M into a matrix, where each row contains a sequence in M, and assume there are no duplicate columns, and that each column has at least one entry that is one.

2.1. Background and Major Combinatorial Tool

Definition 2.1. Two columns (or sites) in M are said to "conflict" if and only if the two columns contain three rows with the pairs $1,1$; $0,1$; and $1,0$. A site is called "conflicted" if it is involved in at least one conflict, and is otherwise called "unconflicted."

Recall that a perfect phylogeny is a phylogenetic network without recombinations. Hence, as a graph, it is a directed rooted tree. The following is the classic necessary and sufficient condition for the existence of a perfect phylogeny deriving a set of sequences M. (See Gusfield 1991, 1997; Semple and Steel 2003; Felsenstein 2004 for discussions of this result.)

Theorem 2.1. *There is a perfect phylogeny deriving M if and only if matrix M contains no conflicted sites. Further, if there is a perfect phylogeny for M and all columns of M are distinct, then there is a unique perfect phylogeny for M, and each edge is labeled by at most one site. If there are identical columns, then the perfect phylogeny is unique up to any ordering given to multiple sites that label the same edge.*

Hence it is the existence of conflicts in M that require a deviation from the perfect phylogeny model, and, in this paper, require recombinations in order to derive a history of M.

We next define the conflict graph and its connected components.

Definition 2.2. The conflict graph G contains one node for each site in M. We label each node of G by the site it represents. Two nodes i and j are connected by an undirected edge if and only if sites i and j conflict. See Figure 2.

Definition 2.3. A *connected component* C of G is a maximal subgraph of G such that for any pair of nodes in C there is at least one path between those nodes in G. A *trivial* connected component has only one node, and no edges, and the site associated with that node is unconflicted.

2.2. Prior Structural Results

The main result established in Gusfield et al. (2003, 2004) is a one-to-one correspondence between the nontrivial connected components of G and the galls in a galled tree. More generally, if a gall in a phylogenetic network for M contains a site from one (nontrivial) connected component C of the conflict graph for M, then it contains all the sites from C, and contains no sites from another (nontrivial) connected component. Further, no gall need to contain any unconflicted sites. That is, if a gall contains an unconflicted site, then that site can be moved off the gall to an edge incident with the gall, without changing the placement of any other sites on the gall. Throughout this paper, we will assume that no gall contains any unconflicted sites.

The algorithmic consequence of these results is that, when constructing a galled tree for M (if there is one), we can focus on each nontrivial connected component separately, knowing that those sites can be placed together on one gall, and that gall will contain no other sites. If M cannot be derived on a galled tree, but can be derived on a phylogenetic network N that contains some galls, then the sites on any gall in N are precisely the sites of one nontrivial connected component. Hence, when M cannot be derived on a

galled tree, each nontrivial connected component of the conflict graph nonetheless contains a set of sites that are candidates to appear together on a gall in a phylogenetic network for M. The key task, then, is to determine if those sites can actually appear together on a gall and, if so, how to arrange those sites on a gall. Therefore in this paper, we restrict attention to a single connected component of the conflict graph and derive a strong necessary condition for those sites to appear on a gall, and an algorithm for determining how they can be arranged on the gall.

2.3. Combinatorial Constraints on Galls

In Gusfield et al. (2003, 2004) we established several technical results that we will need in this paper. We review, without proof, those needed results.

Lemma 2.1. *Let Q be a gall in a phylogenetic network N and v be a node on Q. Define N' as the subnetwork of N consisting of all nodes and edges reachable by directed paths from v, not using any edges in Q. If a site i appears on Q, then the state of site i at every node in N' is the same as at node v.*

Definition 2.4. Let C be a set of sites on a gall Q, and let the matrix $M(C)$ be matrix M restricted to the sites in C, and let $M(C)-0$ be $M(C)$ after removal of any all-zero sequence (if there is one) in $M(C)$. Given a phylogenetic network for M, let $S_v(C)$ denote the sequence labeling node v, restricted to the sites in C.

Lemma 2.1 implies the following:

Corollary 2.1. *A sequence is in $M(C)-0$ if and only if it is the sequence $S_v(C)$ for some node v on Q, where v is not the coalescent node of Q. Stated differently, the node labels at the noncoalescent nodes on Q, restricted to sites in C, are exactly the sequences in $M(C)-0$.*

Corollary 2.1 is important because it says that information about the (interior) node labels on any gall is reflected in some sequences at the leaves, and hence that information is contained in extant sequences. This is a property of galls that does not generalize to every nongall recombination cycle, and is intuitively one of the reasons why problems concerning galls and galled trees have efficient solutions. Because we have assumed that the ancestral sequence is the all-zero sequence, $S_w(C)$ is the all-zero sequence, where w is the coalescent node of C.

Definition 2.5. A node v on a recombination cycle Q is called a *branching node* if there is a directed edge (v, v') where v' is not on Q.

The following theorem from Gusfield et al. (2003, 2004) is the technical key to most of the analysis of the combinatorial structure of galls.

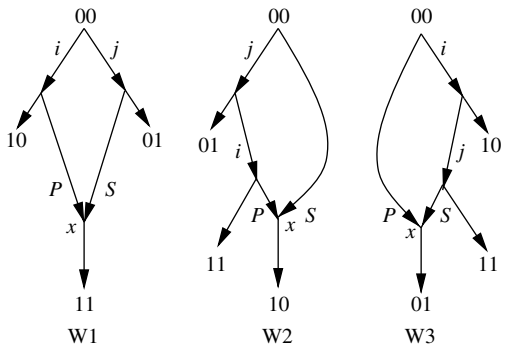

Figure 3 The Three Cases for Theorem 2.2

Note. In each case, the recombination point r_x is between i and j.

Theorem 2.2. *Let N be a phylogenetic network for M. Suppose sites i and $j > i$ are together on some gall Q (with recombination node x) in N. Then sites i and j conflict if and only if the following conditions hold*:

(a) $i < r_x \leq j$.

(b) *Sites i and j are arrayed on Q in one of the following three ways* (*see Figure* 3):

W1: *Site i is on the P side and j is on the S side of Q, and there is a branching node between i and x, and a branching node between j and x. Note*: *in this case, the i, j state pair in the recombinant sequence is* $1, 1$.

W2: *Sites i and j are both on the P side with j above i* (*i.e., j mutates before i does*), *and there is a branching node between j and i, and a branching node between i and x. In this case the i, j state pair in the recombinant sequence is* $1, 0$.

W3: *Sites i and j are both on the S side with i above j, and there is a branching node between i and j, and a branching node between j and x. The state pair in this case is* $0, 1$.

Lemma 2.2. *Let C be a nontrivial connected component of the conflict graph for M, whose sites appear on a gall Q in a phylogenetic network for M. Then C must be bipartite, and the bipartition is unique*: *the* (*indices of the*) *sites on one side of the bipartite graph must be strictly smaller than the sites on the other side.*

Definition 2.6. Given the bipartite graph for C, we call the side containing the smaller sites the L (left) side, and the other side the R (right) side.

It is easy to find the bipartition and select r: let p be the largest (by index) node in C that is connected only to larger nodes in C, and let q be the smallest node in C that is connected only to smaller nodes. Then r can be chosen to be any integer strictly larger than p and less or equal to q, and this defines L and R. The permitted range for r is called the *recombination interval* of C.

3. Arranging the Sites of C on Q

This section begins the development of the new results in this paper. Here we look in detail at how the sites on a connected component C can be arranged on

the gall Q. In Gusfield et al. (2003, 2004) we showed that, except for degenerate cases, the arrangement is unique, and in any case, there are *essentially* at most three ways to arrange the sites. In this paper we go further, by exactly characterizing what those permitted arrangements are, also obtaining a faster algorithm for finding those arrangements.

Definition 3.1. Let N be a fixed phylogenetic network for M, and Q be a gall in N containing the sites on the connected component C of the conflict graph. We define $P_L \subseteq L$ to be all the sites in L that are together on the P side of Q, and define $P_R \subseteq R$ to be all the sites in R that are together on the P side of Q. Similar definitions apply for S_L and S_R.

Note that any one of these sets may be empty, but at most one of P_L or S_L can be empty, and at most one of P_R and S_R can be empty, because both L and R are nonempty.

3.1. The Canonical Arrangement

Suppose we know the sets P_L, P_R, S_L, and S_R on Q in a phylogenetic network N (how we identify these sets will be discussed later). We will characterize all the possible ways that these sites may be arranged on Q. Here we discuss in detail the sites in P_L and P_R. The case for sets S_L and S_R is symmetric.

Definition 3.2. For a site i, ONE(i) is the number of ones in column i.

Lemma 3.1. *Suppose set P_L has at least two sites. On Q, the mutation for a site $i \in P_L$ must occur before (above) the mutation for a site $i' \in P_L$ if and only if* ONE(i) > ONE(i'). *Similarly, the mutation for a site $j \in P_R$ must occur before the mutation for a site $j' \in P_R$ if and only if* ONE(j) > ONE(j').

Proof. Suppose i occurs before i' on the P side of Q. Let v be any node on Q below mutation i, where v is not the recombination node of Q. Let (v, v') be any edge where v' is not on Q. We say that the edge (v, v') *branches off* of Q. Then both i and i' have state one at v', and by Corollary 2.1, every leaf below v' contributes one to both ONE(i) and ONE(i'). But because i occurs before i', and no two columns in M are identical, there must be a branching node between the mutations for i and i'. Then, by Corollary 2.1, the leaves reachable from that branch contribute to ONE(i) but not to ONE(i'). These statements also hold for ONE(j) and ONE(j'). Now consider the recombination node x of Q, and note that the states of both i and i' are one at x, and the states of both j and j' are 0 at x. Hence every leaf below x that contributes to ONE(i) also contributes to ONE(i'), and none of those leaves contribute to either ONE(j) or ONE(j'). This proves that ONE(i) > ONE(i'). Similarly, if i' occurs before i on P, then ONE(i') > ONE(i). Because i either occurs before i', or i' occurs before i, the Lemma is proved. □

Definition 3.3. The *internal arrangement* of P_L refers to the order in which the sites in P_L occur on gall Q, ignoring any sites of P_R, and a similar definition applies for the internal arrangement of P_R.

Lemma 3.1 says that, given P_L, the internal arrangement of P_L is forced, and can be constructively determined, as can the internal arrangement of P_R. Hence, we can unambiguously refer to the kth site of P_L on Q, or the lowest k sites of P_L, etc.

Definition 3.4. Let $F(P_L)$ and $F(P_R)$ be lists containing the sets P_L and P_R ordered by their ONE counts, largest first. These two lists specify the forced internal arrangements of P_L and of P_R on Q.

Because the internal orders of P_L and P_R are fixed, the arrangement of sites on the P side of Q consists of an *interleaving* of the lists $F(P_L)$ and $F(P_R)$. We next examine how P_L and P_R are permitted to interleave on Q. Note that for a pair of sites $i \in P_L$ and $j \in P_R$, the values ONE(i) and ONE(j) do not completely determine the relative order of sites i and j on P. In particular, it can happen that j occurs before i on P, and yet ONE(j) < ONE(i) because the state of site $j \in P_R$ becomes zero at the recombination node x of Q, and remains at zero in the subtree rooted at x, while the state of i remains one at and below x.

Definition 3.5. For any site $j \in P_R$, let k_j denote the number of sites in P_L with which site j conflicts.

In the case that at least one of P_L and P_R is nonempty, we determine an arrangement (which we call the *canonical arrangement*) of P_L and P_R on the P side of Q as follows:

In the order that they are in $F(P_R)$, place each site j in P_R (somewhere) below the previously placed site from $F(P_R)$, and immediately above the lowest k_j sites of P_L. (Note that k_j is at least one, by Theorem 2.2 and the fact that every site in P_R is conflicted.)

By the necessary direction of Theorem 2.2, the canonical arrangement of P_L and P_R is consistent with the conflicts with which P_L and P_R are involved, but in order to comply with the sufficient direction of Theorem 2.2, we must specify where branching nodes and edges occur on P.

3.1.1. Adding in the Branching Nodes. By the assumption that no two columns are identical, there must be a branching edge between any consecutive sites on Q that are both in P_L, and similarly between any consecutive sites that are both in P_R. By construction, if a site $s \in P_R$ is just above a site $t \in P_L$, then s and t conflict (according to C), and so by Theorem 2.2 we need to put a branching node and edge (branching off of Q) between sites s and t. This leaves only the case of two consecutive sites s and t where $s \in P_L$ and $t \in P_R$. We must determine if a branching node and edge should be placed between these two sites.

Note that the last site t' on the P side must be in P_L by Theorem 2.2 and the fact that every site in P_R

is conflicted. It follows that s, t, and t' are three distinct sites arranged in that relative order on the P side of Q. If there were a branching node and edge between s and t, then by Corollary 2.1, there would be a sequence in M with the (s, t, t') state triple $(1, 0, 0)$. Now note that the (s, t, t') state triple is $(1, 1, 0)$ at nodes below t and above t' on the P side; it is $(1, 1, 1)$ at the node below t' and above x on P; and it is $(1, 0, 1)$ at x. Therefore, we should place a branching node and edge between s and t if and only if there is a sequence in M with the (s, t, t') state triple of $(1, 0, 0)$. Finally, as established in Theorem 2.2, a branching node and edge must be placed below the last site on the P side of Q.

The time needed to determine the branching node and edge placements is constant unless s is in P_L and t is in P_R, in which case the time is $O(n)$ for such a consecutive pair, and $O(n|C|)$ for the entire gall Q. The definition of t' as the last site on the P side simplifies the implementation and the achievement of the claimed time bound.

Note that the above rules cover the cases where one of P_L or P_R is empty. Of course, if both are empty, then no sites or branching edges are placed on the P side of Q.

If either of S_L or S_R is nonempty, then we can similarly establish an arrangement of these sets on the S side of Q. The only modification to the method described for the P side, is that the roles of L and R are reversed. Summarizing what has been established so far, we have:

Theorem 3.1. *Let C be a nontrivial connected component of the conflict graph, and let Q be the gall containing those sites. Assuming we know the partition of C into the sets P_L, P_R, S_L, and S_R, the canonical arrangement of those sets creates all and only the conflicts specified in C.*

Proof. By Theorem 2.2, the canonical arrangement places the sites, nodes, and branches on Q in a way that creates the required conflicts between P_R, P_L pairs, and between S_L, S_R pairs. By the assumption that we know the sets P_L, P_R, S_L, and S_R, and the fact that every site in C is in conflict, every P_L, S_R pair must be in conflict in C, and the canonical arrangement creates the conditions for those conflicts as well. Using Theorem 2.2 and the rules of the canonical arrangement, we see that no conflicts are created on Q that are not specified by edges in C. □

Theorem 3.1 says that the canonical arrangement produces the pattern of conflicts and nonconflicts given in C. But, in general, having the same pattern of conflicts is not a sufficient condition for creating the needed sequences. We now address that issue.

Theorem 3.2. *If there is a phylogenetic network N for M where the sites of C appear on a gall Q, and the sites are partitioned into the sets P_L, P_R, S_L and S_R, then there is a phylogenetic network for M where the sites on Q are arranged as in the canonical arrangement. Further, all other arrangements of sites of Q that can be in a phylogenetic network for M and use the same partition, can be created from the canonical arrangement by one or more local switches of neighboring sites that have no branching node between them.*

Proof. Note that most details of the canonical arrangement are *forced* by Theorem 2.2 and Lemma 3.1. Those forced details must be the same as in the assumed arrangement of Q in N. The only nonforced aspect of the canonical arrangement occurs if there is a consecutive s, t pair of sites (with $s \in P_L$ and $t \in P_R$) with no branching node between them. In that case, we can interchange the positions of s and t to create another arrangement that keeps the same pattern of conflicts and nonconflicts, and generates exactly the same sequence labels on the nodes of Q. Further, this interchange is the only change from the canonical arrangement in which either s or t can participate. To see this, note that by the rules for placing branching nodes and edges, either s is the first site on the P side, or there must be a branching node and edge just above s in the canonical arrangement. Similarly, there must be a node and branching edge just below t. So, only the canonical arrangement and the arrangements created from the canonical arrangement by these local switches (if any) are consistent with the pattern of conflicts in C, and the fixed partition of sites into sets P_L, P_R, S_L, and S_R. Moreover, all these arrangements produce the same sequence labels at the nodes of Q. It follows that, if there is a phylogenetic network for M containing a gall with the partition P_L, P_R, S_L, and S_R in Q, then there is a phylogenetic network for M where the sites on Q are arranged as in the canonical arrangement. Further, this gives a characterization of *all* the possible arrangements of those sites on Q, given the partition P_L, P_R, S_L, and S_R. □

In most cases, if there is no branching node and edge between two consecutive sites (one in L and one in R), the specific ordering of those two sites cannot be known and we often consider that two sites on the same edge appear unordered. With that viewpoint, we conclude:

Corollary 3.1. *If multiple sites on the same edge are considered unordered, then a fixed partition of L and R into sets P_L, P_R, S_L, and S_R (along with M), forces the arrangement of sites on Q.*

3.2. Identifying the Permitted Partitions

Definition 3.6. An arrangement of the sites on Q is called *permitted* if it occurs in some phylogenetic network N for M. A partition of the sites in L and

in R into sets P_L, S_L, P_R, and S_R is called *permitted* if it occurs in some permitted arrangement of the sites on Q.

Above, we assumed we knew how the sites were partitioned into sets P_L, P_R, S_L, and S_R, and showed how to arrange those sites on Q. Now we focus on how to identify permitted partitions.

To begin, we establish that for any pair of sites i, i' in L, either i and i' are on the same side of Q in all permitted arrangements of Q, or they are on opposite sides of Q in all permitted arrangements. Moreover, we can determine efficiently which of the two cases hold. Similar statements are true for R.

LEMMA 3.2. *A pair of sites i, i' in L must be on the same side of Q in all permitted arrangements of Q if and only if there is a row in M that has a one for both i and i'.*

PROOF. Suppose i and i' are on the same side of Q, and without loss of generality, suppose mutation i occurs before mutation i'. By Theorem 2.2, there must be a branching edge off Q below the last conflicted site on both sides of Q. At that branching node both sites i and i' are set to one. By Corollary 2.1, there will be a sequence in M where the (i, i') state pair is 1,1.

Conversely, if i and i' are on opposite sides of Q, then at the recombination node x of Q, one parent node of x has (i, i') state pair of $(0, 1)$ and the other parent has state pair $(1, 0)$. Because both i and i' are in L, the recombination point does not occur between i and i' and hence at the recombinant node x the (i, i') state pair must be equal to what it is at the parent of x on the P side of Q. Therefore, the nodes labels on Q will contain only the (i, i') state pairs $(0, 0), (1, 0)$, and $(0, 1)$, and so by Corollary 2.1, no row in M will have a one for both i and i'. Hence, if there is a row in M where both sites i and i' have value one, then both i and i' must be on the same side of Q. □

Therefore, in $O(n)$ time, we can determine if i and i' must be together on one side of Q or on separate sides of Q in any permitted arrangement of Q. By fixing a single site in L as a reference, $|L| - 1$ such checks determine that all of L must be together on one side of Q (in which case we assign L to L_1), or to partition L into two sets (L_1 and L_2) that must be on opposite sides of Q. So in $O(n|L|)$ time, we can uniquely partition the elements of L into sets L_1 and L_2. Similarly, we can examine and uniquely partition the elements of R, in $O(n|R|)$ time into sets R_1 and R_2.

Note that by convention, L_1 and R_1 are never empty, but L_2 and R_2 might be empty.

In each component C, the time to partition L and R into sets L_1, L_2, R_1 and R_2 is $O(n|C|)$.

3.3. Assigning L_1, L_2, R_1, and R_2 to P_L, P_R, S_L, and S_R

Now we develop an efficient algorithm to find all the ways in which L_1, L_2, R_1 and R_2 can be assigned to P_L, S_L, P_R and S_R, and hence all the permitted partitions of L and R. The analysis is broken into a number of cases, depending on whether L_2 or R_2 are nonempty, and whether any of the four sets has only one element. We start with two needed definitions.

DEFINITION 3.7. We let $f(L_1)$ be the site in L_1 whose ONE count is largest over all sites in L_1, and define $l(L_1)$ to be the site in L_1 whose ONE count is smallest over all sites in L_1. Analogous definitions and facts hold for sites $f(L_2)$, $f(R_1)$, $f(R_2)$, as well as for $l(L_2)$, $l(R_1)$, and $l(R_2)$.

Note that all of these defined sites (if they exist) can be efficiently identified. By Lemma 3.1, $f(L_1)$ must be placed on Q before any other site in L_1, and site $l(L_1)$ must be placed on Q after all other sites in L_1.

DEFINITION 3.8. For a given set P_L, we let $f(P_L)$ be the first site (topmost in Q) in P_L, and let $l(P_L)$ be the last site (lowest) in P_L. Analogous definitions hold for $f(P_R)$, $l(P_R)$, $f(S_L)$, $l(S_L)$, $f(S_R)$, and $l(S_R)$.

By Lemma 3.1, these are all well defined (if they exist).

3.3.1.

Case 1. $|L_1|$, $|R_1|$, $|L_2|$, and $|R_2| \geq 1$.

On the assumption that the sites in C appear on a gall in a phylogenetic network N for M, and that each site in C is conflicted, we know from Theorem 2.2 that every site in P_R conflicts with some site in P_L and no other sites; similarly, every site in S_L conflicts with some site in S_R and no other sites. Further, every site in P_L conflicts with every site in S_R, and no site in P_R conflicts with any site in S_L. Therefore, we can find the sets P_R and S_L as follows: For each of the four combinations of p and q, where each p and q is either one or two, take any site i in L_p and any site j in R_q, and determine whether site i conflicts with no sites in R_q, and j conflicts with no sites in L_p. By the above argument, this will be true for exactly one of the four p, q combinations, and for that combination, $P_R = R_q$ and $S_L = L_p$. Knowing which set is P_R and which set is S_L also determines which set is P_L and which is S_R because L_1 and L_2 must be on opposite sides of Q, and R_1 and R_2 must also be on opposite sides.

Note that in Case 1, the time needed to determine the partition is $O(n)$.

3.3.2.

Case 2. $|L_2| = 0$ and $|L_1|, |R_1|, |R_2| \geq 1$.

By Theorem 2.2, the only way that sites in both R_1 and R_2 can be in conflict is for L_1 to be P_L. We must determine whether R_1 is P_R or S_R.

3.3.3.

Case 2a.

We consider first the subcase where $|L_1| \geq 2$. We claim that $R_1 = P_R$ if and only if there is a sequence in M that has 1,1,0 for the sites $f(L_1), l(R_1), l(L_1)$, respectively.

To prove this, suppose first that $R_1 = P_R$. Note that $l(P_R)$ must be above $l(P_L)$ and there must be a branching node between them for site $l(P_R)$ to be conflicted. There must also be a branching node between $f(P_L)$ and $l(P_L)$ because otherwise their columns would be identical in M. Now $l(P_R)$ may or may not be above $f(P_L)$, but in either case, at the branching node just below the lower of those two sites the $(f(P_L), l(P_R), l(P_L))$ state triple will be 1,1,0, and by Corollary 2.1, there will be a sequence in M with that state triple. However $f(P_L)$ is $f(L_1)$ and $l(P_L)$ is $l(L_1)$, and on the assumption that $R_1 = P_R$, $l(P_R)$ is $l(R_1)$, so this part of the proof is complete.

Conversely, if $R_1 = S_R$, then $f(L_1)$ and $l(R_1)$ can both have state one only at the recombination node for Q, where $l(L_1)$ also has state one. So by Corollary 2.1, there will be no sequence in M with the required state triple.

Note that the time needed to implement Case 2a is $O(n)$.

3.3.4.

Case 2b.

Now consider the subcase where $|L_1| = 1$. Clearly, every site in R conflicts with the single site in P_L, and by Theorem 2.2, all sites in P_R must be above the single site in P_L. In this case, either set R_1 or R_2 could be assigned to P_R, and the other set assigned to S_R. That is, the data at the conflicted sites in C do not exclude either possibility. In this case, there are two permitted partitions of R into P_R and S_R. The time to find these is constant.

3.3.5.

Case 3. $|R_2| = 0$ and $|R_1|, |L_1|, |L_2| \geq 1$.

This is symmetric to Case 2, and left to the reader.

3.3.6.

Case 4. $|L_2| = |R_2| = 0$ and $|L_1|, |R_1| \geq 1$.

By Theorem 2.2 and Corollary 3.1, L_1 and R_1 can be assigned in only three possible ways, as follows:

(i) $L_1 = P_L$ and $R_1 = P_R$ (interleaved on the P side as in the canonical arrangement).

(ii) $L_1 = S_L$ and $R_1 = S_R$ (interleaved on the S side as in the canonical arrangement).

(iii) $L_1 = P_L$ and $R_1 = S_R$.

We will show now that when both L_1 and R_1 have at least two sites, only one of these three assignments is possible and we can efficiently determine which one. However, if one of the sets has a single site and the other has at least two sites, then at most two assignments are possible, and if both sets have only a single site, then all three assignments are possible.

3.3.7.

Case 4a.

$|L_1|$ and $|R_1| \geq 2$. Arrangement (i) implies that there is a sequence in M where sites $l(R_1)$ and $f(L_1)$ are both one, but site $l(L_1)$ is zero. However, such a sequence could not exist if the sites are assigned as in (ii) or (iii). To exclude (ii), note that $l(S_L)$ must appear before $l(S_R)$ because $l(S_L)$ is conflicted, so $l(S_L)$ cannot be zero while $l(S_R)$ is one except at the recombination node. But at the recombination node, both $f(S_L)$ and $l(S_L)$ will be zero. To exclude (iii), note that $f(S_R)$ and $f(P_L)$ can be only at the recombination node, where $l(P_L)$ must also be one. So, under the assumption that the sites of C appear on a gall Q, L_1 and R_1 must be assigned as in assignment (i) if and only if there is such a sequence in M.

Symmetrically, L_1 and R_1 must be assigned as in assignment (ii) if and only if there is a sequence in M where $l(L_1)$ and $f(R_1)$ are both one, but $l(R_1)$ is zero.

If neither such sequence is in M, then L_1 and R_1 must be assigned as in assignment (iii). Note that the time to determine the assignment is $O(n)$.

3.3.8.

Case 4b.

$|L_1| = 1$ but $|R_1| \geq 2$.

Only assignment (ii) is possible if there is a sequence in M that has 1,1,0 for $L_1, f(R_1), l(R_1)$. Otherwise, assignments (i) and (iii) are both possible. The time for this case is $O(n)$.

The case when $|L_1| \geq 2$ but $|R_1| = 1$ is symmetric.

3.3.9.

Case 5.

$|L_1| = |R_1| = 1$. In this case, the data from the sites in C do not exclude any of the three assignments detailed in Theorem 2.2.

Because neither L_1 nor R_1 is empty, we have covered all the possibilities, and have detailed all the ways in which the sites of C can be partitioned into the sets P_L, S_L, P_R, and S_R. In summary, we have established the following:

THEOREM 3.3. *If the sites in C can be arranged on a gall in a phylogenetic network, there are at most three permitted partitions of L and R into sets P_L, S_L, P_R, and S_R, and we can find these partitions in $O(n)$ time.*

PROOF. The partition of L into two sets L_1 and L_2 was forced, except for the names of the two sets. The same holds for the partition of R into R_1 and R_2. The analysis of how L_1, L_2, R_1, and R_2 could be assigned to P_L, S_L, P_R, and S_R showed that there were at most three possible assignments. □

Theorem 3.3 bounds the number of permitted arrangements from above, but it doesn't establish that each of the partitions discussed in the five cases above (when the case applies) actually leads to a permitted arrangement, if there is one. We now establish that fact.

THEOREM 3.4. *Let C be a connected component of the conflict graph. If there is a phylogenetic network N for M where the sites of C appear on a gall, then each partition*

enumerated in the five cases is a permitted partition, i.e., when the appropriate case applies, the partition detailed in that case can be used to create a permitted arrangement of the sites on Q.

Proof. The claim is trivially true when the partition is forced, as in cases 1, 2a, 3, and 4a. However in cases 2b, 4b, and 5, the partition is not forced, and we must show that each partition enumerated in those cases is a permitted partition. We will prove this for Case 2b in detail. The other cases are similar, and left to the reader.

Suppose Case 2b applies, and suppose that in the arrangement of conflicted sites on Q, the sites in R_1 are above the single L_1 site on the P side of Q, and the sites in R_2 are on the S side. Let v be the branching node just below the L_1 site, and let A denote the directed subnetwork hanging off of Q at node v. Let B denote the subnetwork hanging off the recombination node x of Q. If we now move the L_1 site to below the R_2 sites, but above the last branching node on the S side, and reassign that side to be the P side (and the other side the S side), and we exchange the subnetworks A and B (i.e., hanging A off of x, and B off of the node just below L_1), the resulting network will again be a phylogenetic network for M. In the new arrangement, set R_2 is above the L_1 site on the P side, and set R_1 is on the S side. Thus, there is a phylogenetic network for both assignments enumerated in Case 2b. By symmetry, the transformation can be made in the opposite direction as well, and hence the theorem is proved when Case 2b applies. □

Theorem 3.5. *If the c sites on a connected component C can be arranged on a gall in a phylogenetic network, then as long as multiple sites on the same edge are unordered, the sites can be arranged in at most three ways, and we can find those arrangements in $O(cn + c\log c)$ time.*

Proof. Corollary 3.1 established that the canonical arrangement of the sites on Q was forced given the partition of sites into sets P_L, S_L, P_R, and S_R. The time to implement the canonical arrangement is $O(cn)$. Theorem 3.3 established that L and R can be partitioned in at most three ways. The time to establish the partitions is $O(n)$. Both tasks assumed that the sites in L and R were respectively sorted by their ONE counts. That requires $O(c\log c)$ time, although in a practical implementation we would sort all the sites of M first to get a better overall time bound for arranging all the galls. □

4. Bi-Convexity

We now establish a strong necessary structure that a connected component of the conflict graph must have in order for its sites to be arranged on a gall in a phylogenetic network.

Definition 4.1. A bipartite graph with node sets A and B is called *convex for B* if the nodes of B can be ordered so that, for each node $i \in A$, the set of nodes in B to which i is adjacent form a closed interval in B. That is, i is adjacent to j and $j' > j$ in B if and only if i is adjacent to all nodes in the closed interval $[j, j']$. A bipartite graph is called *bi-convex* if sets A and B can be ordered so that it is simultaneously convex for A and convex for B.

Bi-convex graphs are perfect graphs and are a subset of the chordal bipartite graphs (Dragan 2000), which have structure that can be exploited to develop particularly efficient algorithms for problems that are otherwise NP-hard. We are interested in bi-convex graphs partly because of such speed-ups, but mostly because we are interested in how structures in the conflict graph influence structures in a phylogenetic network.

Theorem 4.1. *Let G be a conflict graph for a matrix M that can be derived on a galled tree. Then any connected component C in G is bi-convex, and so G is also bi-convex. More generally, if C is a connected component of G and the sites on C appear on a gall in some phylogenetic network for M, then C must be bi-convex.*

Proof. Let Q be the gall corresponding to a component C. The pairwise conflicts created by the canonical arrangement of sites on Q are exactly the conflicts specified by the edges in C, so we can use the canonical arrangement to prove the theorem. As already noted, C must be bipartite, with nodes in $L = P_L \cup S_L$ on one side and $R = P_R \cup S_R$ on the other side of C.

Let BC be the bipartite graph obtained from C by ordering the nodes of C as follows. In each set P_L, P_R, S_L, and S_R that is nonempty, order the nodes according to their ONE values, largest first; then place the nodes of S_R before the nodes of P_R, and place the nodes of P_L before the nodes of S_L. In the canonical arrangement of sites on Q, each site i in P_L conflicts with every site in S_R, and with every site in P_R that is above it in the canonical arrangement. Therefore node i in S_L is adjacent to every node in S_R, and some initial interval of nodes in P_R, so the set of nodes adjacent to i form a closed interval of nodes. Similarly, each node i' in S_L is adjacent to some ending interval of nodes in S_R and to no nodes in P_R, so the set of nodes adjacent to i' form a closed interval. Hence BC is convex for R. Each node j in P_R is adjacent to some ending interval of nodes in P_L and to no others; and each node j' in S_R is adjacent to every node in P_L and to an initial interval of nodes in S_L, so BC is convex for S. Hence, C is bi-convex. □

4.1. Site Consistency

A *node cover* of a graph is a set of nodes V such that every edge in the graph touches at least one node in V. A *minimum* node cover is a node cover with the

fewest nodes. It is easy to see that the minimum number of columns to remove from M so that no conflicts remain is given by a minimum node cover of the conflict graph. This is called the *site-consistency* problem, and it is NP-hard in general (Day and Sankoff 1986). However, the node cover problem can be solved in polynomial time (by network flow) on any bipartite graph, and in linear time when the graph is chordal bipartite, given some additional information about the graph (Dragan 2000). This approach gives a solution to the site-consistency problem that is faster than by network flow, for chordal bipartite graphs, and we conjecture that the problem can be solved in linear time, when M can be derived on a galled tree.

Acknowledgments

The authors would like to thank the National Science Foundation for support for this work under award EIA-0220154.

References

Day, W. H., D. Sankoff. 1986. Computational complexity of inferring phylogenies by compatibility. *Systematic Zoology* **35** 224–229.

Dragan, F. 2000. Strongly orderable graphs: A common generalization of strongly chordal and chordal bipartite graphs. *Discrete Appl. Math.* **99** 427–442.

Felsenstein, J. 2004. *Inferring Phylogenies*. Sinauer, Sunderland, MA.

Gusfield, D. 1991. Efficient algorithms for inferring evolutionary history. *Networks* **21** 19–28.

Gusfield, D. 1997. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge, UK.

Gusfield, D. 2004. Optimal, efficient reconstruction of root-unknown phylogenetic networks with constrained recombination. Technical report, Department of Computer Science, University of California, Davis, CA.

Gusfield, D., S. Eddhu, C. Langley. 2003. Efficient reconstruction of phylogenetic networks (of SNPs) with constrained recombination. *Proc. Second CSB Bioinformatics Conf.* IEEE Press, Los Alamitos, CA.

Gusfield, D., S. Eddhu, C. Langley. 2004. Optimal, efficient reconstruction of phylogenetic networks with constrained recombination. *J. Bioinformatics Comput. Biol.* **2** 173–213.

Hein, J. 1990. Reconstructing evolution of sequences subject to recombination using parsimony. *Math. Biosci.* **98** 185–200.

Hein, J. 1993. A heuristic method to reconstruct the history of sequences subject to recombination. *J. Molecular Evolution* **36** 396–405.

Kececioglu, J. D., D. Gusfield. 1998. Reconstructing a history of recombinations from a set of sequences. *Discrete Appl. Math.* **88** 239–260.

Myers, S. R., R. C. Griffiths. 2003. Bounds on the minimum number of recombination events in a sample history. *Genetics* **163** 375–394.

Norborg, M., S. Tavare. 2002. Linkage disequilibrium: What history has to tell us. *Trends Genetics* **18** 83–90.

Posada, D., K. Crandall. 2001. Intraspecific gene genealogies: Trees grafting into networks. *Trends Ecology Evolution* **16** 37–45.

Schierup, M. H., J. Hein. 2000. Consequences of recombination on traditional phylogenetic analysis. *Genetics* **156** 879–891.

Semple, C., M. Steel. 2003. *Phylogenetics*. Oxford University Press, Oxford, UK.

Song, Y., J. Hein. 2003. Parsimonious reconstruction of sequence evolution and haplotype blocks: Finding the minimum number of recombination events. *Proc. 2003 Workshop Algorithms Bioinformatics*, *Lecture Notes in Computer Science*, No. 2812, Springer-Verlag, Berlin, Germany, 287–302.

Wang, L., K. Zhang, L. Zhang. 2001. Perfect phylogenetic networks with recombination. *J. Comput. Biol.* **8** 69–78.