

Efficient Reconstruction of Phylogenetic Networks (of SNPs) with Constrained
Recombination

by

Satish Eddhu

B.E. (Bharathiar University, India)

Thesis

Submitted in Partial Satisfaction of the Requirements for the Degree of

Master of Science

in

COMPUTER SCIENCE

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Committee in Charge

2003

Efficient Reconstruction of Phylogenetic Networks (of SNPs) with Constrained Recombination

Abstract

A phylogenetic network is a generalization of a phylogenetic tree, allowing structural properties that are not tree-like. With the growth of genomic data, much of which does not fit ideal 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 [10]. However, to date, very little has been published on this, with the notable exception of the paper by Wang et al.[10] which is the model we improve on. Other related papers include [4, 5, 8]. In [4, 5] the solution is a set of trees, one for each of the sequence segments and a recombination history that relates the trees. The model in [8] computes the minimum cost recombination history with two of the input sequences as prototype pairs and all other sequences produced by exactly one recombination.

Wang et al. [10] studied the problem of constructing a phylogenetic network for a set of n binary sequences derived from the all-0 ancestral sequence, when each site in the sequence can change from 0 to 1 at most once in the network, and recombination between sequences is allowed. They showed that the problem of finding a phylogenetic network that minimizes the number of recombination events is NP-hard, but gave a polynomial-time algorithm ($O(nm + n^4)$ -time, for n sequences of length m each) that was intended to determine whether the sequences could be derived on a phylogenetic network where the recombination cycles are node disjoint. In this paper, we call such a phylogenetic network a "galled-tree". Unfortunately, the algorithm in [10] is incomplete and does not constitute a necessary test for the existence of a galled-tree for the data. We show this by example. Through simulations, we observed that the algorithm of [10] failed on 5-65% of (coalescent derived) sequence sets that can actually be derived on galled-tree.

By more deeply analyzing the combinatorial constraints on cycle-disjoint phylogenetic networks, we obtain an algorithm that runs in $O(nm + n^3)$ -time and is guaranteed to be both a necessary and sufficient test for the existence of a galled-tree for the data. If there is a galled-tree, the algorithm constructs one, and obtains an implicit representation of all the galled trees for the data, and can create these in linear time for each one. We also show two additional results related to galled trees: first, any set of sequences that can be derived on a galled tree can be derived on a true tree (without recombination cycles), where at most one back mutation is allowed per site; second, the site compatibility problem (which is NP-hard in general) can be solved in linear time for any set of sequences that can be derived on a galled tree.

The combinatorial constraints we develop apply (for the most part) to node-disjoint cycles in any phylogenetic network (not just galled trees), and can be used, for example, to prove that a given site cannot be on a node-disjoint cycle in any phylogenetic network. Perhaps more important than the specific results about galled-trees, we introduce an approach that can be used to study recombination in phylogenetic networks that go beyond galled-trees.

Contents

1	Introduction to Phylogenetic Networks and Galled-Trees	1
1.1	A formal definition of phylogenetic networks	1
1.2	Which Phylogenetic Networks are Biologically Informative?	4
1.2.1	Galled-trees: A Biological and Algorithmically Motivated Structural Restriction	4
2	Combinatorial Definitions and Observations	5
2.1	Combinatorial Background and Major Combinatorial Tool	6
2.2	Major Tool: The Conflict Graph and its Connected Components	6
2.3	Combinatorial constraints on galls	7
2.4	Unconflicted Sites need not be on Galls	12
3	Arranging the Gall Q_c	16
3.1	Selecting the Recombination Point r on Q_c	16
3.2	Arranging the Sites of C on Q_c	17
3.2.1	Initial Delineation of Conflicted Sites in Q_c	18
3.2.2	Case 1: $ L_1 , R_1 , L_2 $ and $ R_2 \geq 1$	19
3.2.3	Case 2: $ L_2 = \emptyset$ and $ L_1 , R_1 , R_2 \geq 1$	20
3.2.4	Case 3: $ L_2 = R_2 = \emptyset$ and $ L_1 , R_1 \geq 1$	21
3.2.5	Interleaving Sites on a Side of Gall	21
4	Connecting the Galls in a Galled-tree	23
5	Adding the Leaf Sequences and Unconflicted Sites	25
5.1	Constructing the Top Perfect Phylogeny in a Galled-tree	25
5.2	Connecting the Gall Forest to the Top Perfect Phylogeny	26
5.3	Constructing Subnetworks Hanging off of Galls	28
5.4	Handling Duplicate Sites	30

6	The Final Check	30
7	Results	31
8	Relation to the Back-mutation Model	35
9	One Provable Lower Bound on m_M	36
10	Future Work and Open Questions	37
11	Acknowledgment	37

1 Introduction to Phylogenetic Networks and Galled-Trees

With the growth of genomic data, much of which does not fit ideal 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 a greater need to understand the algorithmics and combinatorics of phylogenetic networks. A phylogenetic network is a tree-like structure that describes the evolutionary history of a set of sequences based on probable historical events such as gene mutation and recombination. It is formally defined in the next subsection. 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, to date, very little has been published on phylogenetic networks, with the notable exception of the paper by Wang et al.[10]. It is the model that we improve on.

1.1 A formal definition of phylogenetic networks

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-0 for convenience); that for any site in the sequences, there is exactly one point in the history where that state of that site mutates (due to a point-mutation) from 0 to 1; and that two sequences are permitted to recombine in an 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 set threshold. With these definitions, a classic perfect phylogeny is a phylogenetic network which is topologically a directed, rooted tree, i.e., lacking any cycles in the underlying (undirected) graph.

There are four components needed to specify a phylogenetic network: a directed acyclic graph; an assignment of mutations or sites (integers) to edges; an assignment of a sequence to each non-recombination node; an assignment of a sequence to each recombination node. We will define each

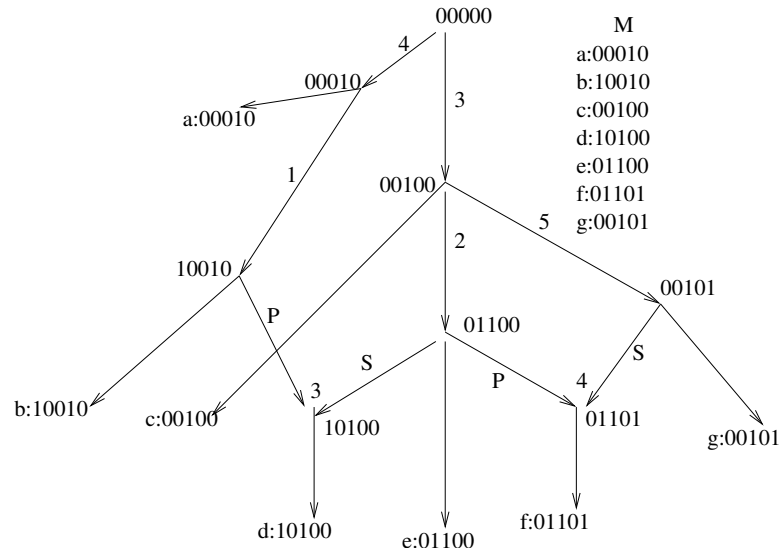


Figure 1: A phylogenetic network N with two recombination nodes. The matrix of sequences M that are derived by N is shown at the right. Note that the node with sequence label 01100 is sequence S for the left recombination node, and is sequence P for the right recombination node. The recombination points are 2 and 4 for the left and right recombination nodes respectively.

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.

Each node in N is labeled by an m -length binary sequence, starting with the root node which is labeled with the all-0 sequence. Since 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 non-root nodes, in order of their appearance in the list, as follows:

- a) For a non-recombination node v , let e be the single edge coming into v . Then the sequence labeling v is obtained from the sequence labeling v 's parent by changing from

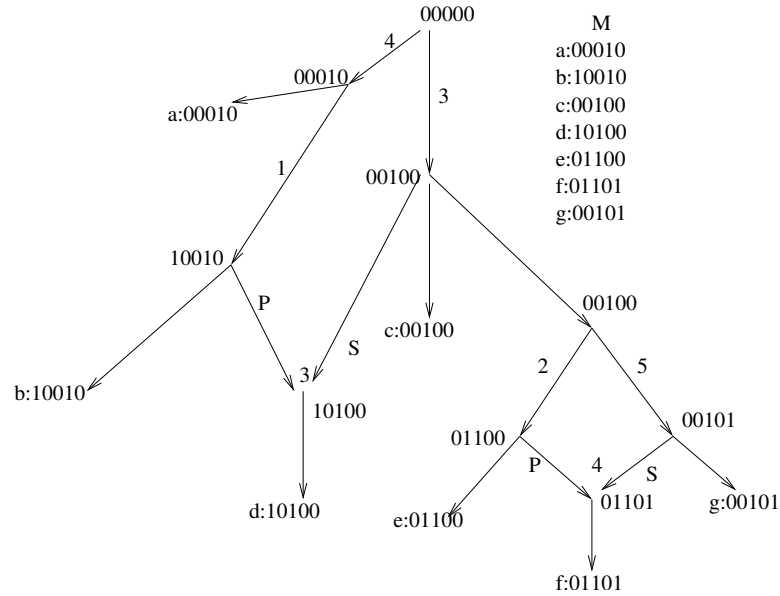


Figure 2: A galled-tree generating the same sequences as the phylogenetic network in Figure 1. Unlike the example shown here, in general the recombinant sequence exiting a gall may be on a path that reaches another gall.

0 to 1 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 is associated with an integer r_x (denoted r , when x is clear by context) between 2 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".

The sequences labeling the leaves of N are the extant sequences, i.e., the sequences that can be observed.

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 . We use the terms "site" and "column" interchangeably.

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. We call that number m_M . Wang et al. [10] showed that the general problem of computing m_M is NP-hard, and Hudson and Kaplan [7] and Myers and Griffiths [9] give combinatorial methods for computing lower-bounds on m_M .

1.2.1 Galled-trees: A Biological and Algorithmically Motivated Structural Restriction

Given the NP-hardness of the problem of computing m_M , Wang et al. suggested a structural restriction on the permitted phylogenetic networks which has both biological and algorithmic appeal. We make the following definitions for clarity of exposition. “““““

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 .*

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.*

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

A galled-tree defines a phylogenetic history where the recombination cycles are node-disjoint, using at most $m/2$ recombinations. A phylogenetic network is likely to be a galled-tree if the level of recombination is moderate, or if most of the observable recombinations are recent. In human

populations, both conditions are believed to hold. Further motivation for galled-trees comes from the fact that if M can be derived by a galled-tree, then it can be derived by a true tree (no underlying undirected cycles) with at most one back mutation per site. A tree with limited back mutations is another model of interest that deviates from the perfect phylogeny model.

Galled-Tree Problem: Given a set M of n binary sequences, each of length m , determine if there exists a galled-tree T that derives M , and if there is one, construct one. Wang et al [10] give an $O(nm + n^4)$ -time algorithm that was intended to solve the Galled-Tree problem. This work is seminal as it is the first paper to introduce a biologically motivated structural restriction for a phylogenetic network that allows a polynomial time algorithm. Unfortunately, the algorithm in [10] is incorrect, and only provides a sufficient condition for the existence of a galled-tree for M .

Main Result: Here we develop a faster $O(nm + n^3)$ -time algorithm that completely solves the Galled-Tree problem. We also show that if there is a galled-tree for M , then all galled-trees for M use the same number of recombinations. It has been proved that the number of recombinations is actually m_M , but the proof is beyond the scope of this paper. Our "canonical" solution is "essentially unique", minimizes the number of sites on the recombination cycles, and can be used to count and produce all the galled-trees for M . In obtaining these results, we develop combinatorial constraints that apply to galls in any phylogenetic network(whether a galled-tree or not). This is useful as a first step in understanding phylogenetic networks in general, and for specific tasks, such as proving that a given site cannot be on any gall in any phylogenetic network. We also show that the problem of removing the minimum number of sites of M , so that the remaining sites have a perfect phylogeny(an NP-hard problem in general) can be solved in linear time.

2 Combinatorial Definitions and Observations

We organize M into a matrix, where each row contains a sequence from M . We also assume for simplicity of exposition that there are no duplicate columns, and that each column has at least one entry that is 1. After we detail the construction of galled trees, we will show that duplicate columns still allow the construction of galled trees and only add an additional constraint to the

conditions necessary.

2.1 Combinatorial Background and Major Combinatorial Tool

Definition 2.1: *Two columns (or sites) in M are said to "conflict" if and only if there are three rows with the pairs 1,1; 0,1; and 1,0 as the values of those columns. A site involved in at least one conflict is called "conflicted", and is otherwise "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 [1] for an exposition.

Theorem 2.1: *There is a perfect phylogeny generating M if and only if the matrix M contains no conflicting 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 the 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 .

2.2 Major Tool: The Conflict Graph and its Connected Components

The central contribution of this paper is to observe that there is combinatorial structure in the pattern of conflicts between columns, and that this structure can be represented and exploited to obtain insights about recombination in phylogenetic networks. We now introduce the *conflict graph*, which represents and exposes some of the combinatorial structure.

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 3.*

Overview: The connected components of G are particularly important. We will show that there is a one-to-one correspondence between the non-trivial connected components of G and the

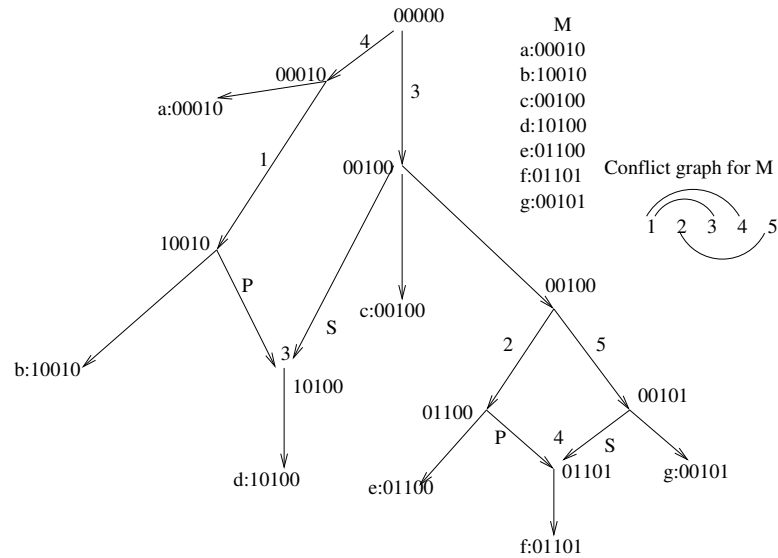


Figure 3: Same as figure 2 but along with the conflict graph M . Site 1 conflicts with the sites 3 and 4, while site 2 conflicts with site 5.

galls in a galled-tree: more generally, every gall in any phylogenetic network contains all the sites of one (non-trivial) connected component, and contains no sites from another (non-trivial) connected component. Furthermore, no gall need contain any unconflicted sites. It follows that every galled-tree for M uses the same number of galls, and the same number of recombinations.

2.3 Combinatorial constraints on galls

In order to prove the claims made in the overview, we next begin an examination of the combinatorial constraints on galls and galled-trees.

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 , i.e., the maximal subnetwork branching off of Q at v . Let i be a site that mutates on an edge along the path from the root to node v . Then the state of site i at every node in N' is the same as at node v .*

Proof: Suppose that at some node in N' , the state of i is different than it is at v . Let x' be such a node with the property that at every ancestor of x' in N' , the state of i is the same as at node v . Since i only mutates once (on Q), the state of i cannot change in N' due to mutation, and

can therefore only change due to recombination. Hence x' must be a recombination node. Now if both parents of x' were in N' , then by choice of x' , the state of i at both parents would be the same as the state at v , and that state would be unchanged at x' regardless of where the recombination point r_x is. So one of the parents of x' , call it p , must be outside of N' . Now consider a path from p back toward the root, and let w be the first ancestor of v reached on this path. Note that w could be v , and in that case, the path also intersects a descendant of v on Q . But the path from w to x' through p , together with the path from w to x' through v , forms a recombination cycle that shares at least one edge with Q , contradicting the assumption that Q is a gall. So the state of site i at every node in N' must be the same as at node v . ■

Definition 2.3: 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 . 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 $M(C)$ if and only if it is the sequence $S_v(C)$ for some node v on Q . Stated differently, the node labels at nodes on Q , restricted to sites in C , are exactly the sequences in $M(C)$.

Proof: The sequences in $M(C)$ are the sequences labeling the leaves of N , restricted to C . If a leaf z is reachable from a node v in Q , not using an edge in Q , then by Lemma 2.1, $S_z(C)$ and $S_v(C)$ are the same. If leaf z is not reachable from any node v in Q , then it must have state 0 for every site i that mutates on Q . In that case $S_z(C)$ is all zeros, which is $S_w(C)$, where w is the coalescent node of Q . ■

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 is contained in extant sequences. This is a property of galls that does not generalize to every non-gall recombination cycle, and is intuitively one of the reasons why problems concerning galls and galled-trees have efficient solutions.

Definition 2.4 A node v on a recombination cycle Q is called a "branching node" if there is a

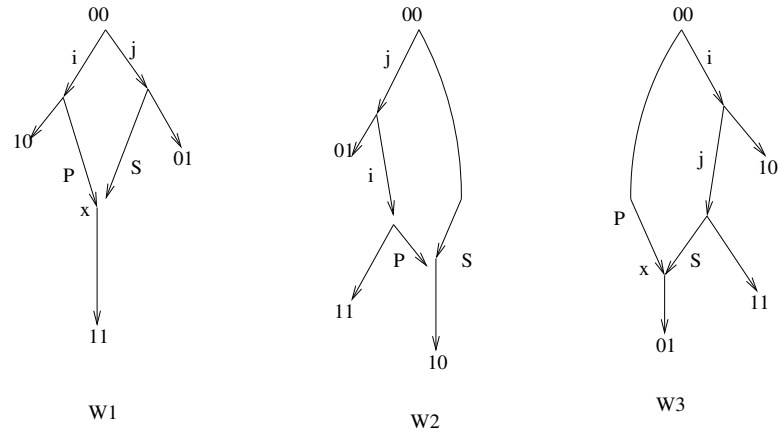


Figure 4: The three cases for Theorem 2.2. In each case, the recombination point x_r is between i and j .

directed edge (v, v') where v' is not on Q .

Theorem 2.2: Let T be a galled-tree for matrix M . Two sites i and $j > i$ (site j occurs after site i) in M conflict if and only if the following conditions hold:

1. i and j are together on the same gall (call it Q) in T , with recombination node x , and $i < r_x \leq j$.
2. Sites i and j are arrayed on Q in one of the following three ways:
 - (a) 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.
 - (b) 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.
 - (c) 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.

The algorithm in [10] is only a sufficient test for the existence of a galled-tree that explains M , because it implicitly assumes that a pair of sites can conflict only due to arrangement W1.

Equivalently, the algorithm in [10] correctly determines whether or not the input sequences can be generated on a galled-tree T having the added constraint: for each site i , if site i mutates on an edge e , then the state of i remains set at 1 at all nodes which are reachable from the end of e . Hence once the state of i mutates from 0 to 1, it never returns to 0, even if recombination happens later on. That is a severe restriction compared to what is allowed by the general definition of a galled-tree. In the galled tree in figure 2, the state of site 4 mutates from 0 to 1, but then is returned to 0 through recombination in the gall shown on the left.

Theorem 2.3: *For any non-trivial connected component C of the conflict graph, and any galled tree T for M , all the sites in C must occur together on a single gall in T .*

This follows by transitivity from the necessary direction of part a) of Theorem 2.2, and the fact that for any pair of sites i and j in C , there must be a path connecting i to j in C .

The following theorem is the complement to Theorem 2.3. It greatly constrains the structure of any galled tree for M and simplifies the development of an efficient algorithm to find a galled tree for M .

Theorem 2.4: *Let T be a galled-tree for M . If sites i and i' are on different non-trivial connected components of the conflict graph, then they must appear on different galls of T .*

We will prove Theorem 2.4 by using the following lemma.

Lemma 2.2: *Let Q be a gall in T with recombination node x , and recombination point r , and let i, i', j, j' be sites on Q , where i conflicts with $j > i$ and i' conflicts with $j' > i'$. Then either i conflicts with j' , or i' conflicts with j .*

Proof: First, by Theorem 2.2, part a) i, i' must both be smaller than the recombination point r of Q , and j, j' must both be greater or equal to r .

Using Theorem 2.2, we consider the three ways that i and j can be arrayed on Q . In each case we will show that either i', j must conflict, or i, j' must conflict.

Case 1: Sites i and j are arrayed as in W1, so site i is on the P-side, and site j is on the S-side of Q , and there are branching edges below each, but above x . Sites i' and j' conflict, so no matter where they are on Q , there must be a branching edge below i' and one below j' , but above x . Now

to avoid conflict with j , site i' must be on the S -side of Q . But then to create conflict between i' and j' , site j' must also be on the S -side. That puts i on the P -side and j' on the S -side, with branching edges below each, and hence by the sufficient direction of Theorem 2.2 case W1, sites i and j' must conflict.

Case 2: Sites i and j are arrayed as in W2, so i and j are both on the P -side, with j above i , and there is a branching edge between them. Since j' conflicts with i' , there must be a branching edge below j' . Hence, by Theorem 2.2, to avoid conflict with i , site j' must be on the P -side, either below i , or above i but below the branching edge between j and i . Either way, j' must be below j on the P -side. But then by Theorem 2.2, because i' and j' conflict, site i' must be on the P -side below j' , and there must be a branching edge below i' . Therefore, j and i' are arrayed on Q as in case W2, and hence by Theorem 2.3 they conflict.

Case 3: Sites i and j are arrayed as in W3, so both are on the on the S -side, with i above j , and there is a branching edge between them. Since i' conflicts with j' , there must be a branching edge below i' . Hence, by Theorem 2.2, to avoid conflict with j , site i' must be on the S -side, either below j , or above j but below the branching edge between i and j . Either way, i' must be below i on the S -side. But then by Theorem 2.2, because i' and j' conflict, site j' must be on the S -side below i' , and there must be a branching edge below j' . Therefore, i and j' are arrayed on Q as in case W3, and hence they conflict. ■

Proof: As per theorem 2.2, sites that conflict with each other must occur on the same gall. Also, lemma 2.2 states that all conflicting sites that occur on a gall must belong to the same connected component. Thus, if sites i and i' are on different non-trivial connected components, they cannot occur on the same gall but must each occur on some gall. So they must occur on different galls. ■

Theorems 2.3 and 2.4 together imply a one-to-one correspondence between the non-trivial connected components of the conflict graph and the galls in a galled-tree: more generally, every gall in any phylogenetic network contains all the sites of one (non-trivial) connected component, and contains no sites from another (non-trivial) connected component.

So far, we have only addressed conflicted sites on a gall. The next section addresses unconflicted sites.

2.4 Unconflicted Sites need not be on Galls

We assume that every gall contains some conflicted sites, since the absence of conflicted sites implies that recombination (and hence a gall) is not necessary to generate the sequences.

Lemma 2.3: *Let N be a phylogenetic network deriving M with a gall Q that contains no conflicted sites on its S -side. Then there is a phylogenetic network N' deriving M with gall Q replaced by gall Q' whose S -side contains no sites and no branching nodes. A symmetric result holds if the P -side contains no conflicted sites.*

Proof: Let P_L denote the set of conflicted sites in Q with index less than r and on the prefix side of Q and P_R denote the set of conflicted sites in Q with index greater than or equal to r and on the prefix side of Q . S_L and S_R are similarly defined for the suffix side. Since all conflicted sites are on the P -side, P_L and P_R are non-empty. Moreover, by Theorem 2.2, there must be a branching edge below the last conflicted site on P . Let r be the recombination point on Q , and define L_r to be all the sites on Q with index less than r , and R_r to be all the sites on Q with index equal to or greater than r .

Let s be an unconflicted site in R_r on the S -side and suppose there is another site t below s . If t is in R_r , then there would have to be a branching node between them in order for the columns in M for s and t to be different. But then s would be in conflict with any conflicted site in P_L , a contradiction. If t is in L_r , but there is a branching node below t , then there is a branching node below s and again s would be in conflict with any site in P_L . But if there is no branching node below t , and since the value of t is set to 0 at the recombination node of Q , the column for t must be all zeros, a contradiction. Hence, if S -side has only unconflicted sites and contains a site s in R_r , s must be the last site on S , and it must not have a branching node below it on Q .

Now, the value of site s is 1 at the recombination node x of Q . If x has a single edge out of it, we can move s off of Q and place it on that single edge out of x . If x has more than one edge out

of it, we create a new, single edge e out of x attached to all the edges that had previously been out of x . We then move s to e , creating a modified phylogenetic network that derives M .

After the removal of site s from Q , the only remaining sites (if any) on the S -side of Q are in L_r . If there are none, then the transformation is complete. Otherwise, all of those sites have value 0 at the recombination node x , so the sequence label at x is maintained even if we remove all those sites from Q . But we need to maintain the sequence label of any branching node v on the S -side, and the sub-graph branching off of Q at v . Hence, we can modify Q as follows: remove the edge into x on the S -side, and add a new edge directly from the coalescent node w of Q to node x . The result is a modified phylogenetic network which derives M , where the S -side of the modified gall Q has no sites and no nodes.

In the transformation above, the part of the old S -side of Q that contains all the sites and nodes, is now a path branching from the coalescent node w . For the modifications of Q used to prove the next theorem, it is useful to maintain the assertion that there is no edge branching off Q at w . So we further modify Q as follows. Let $e = (w', w)$ be the edge into w . Expand e into two edges (w', w'') and (w'', w) by inserting a new node w'' on e , below any sites that are on e . This creates a new edge into w with no sites. Then disconnect any branching edge out of w and reconnect it to w'' . The resulting phylogenetic network clearly derives exactly the same sequences as before. ■

Theorem 2.5: *Let N be a phylogenetic network deriving M with a gall that contains both unconflicted and conflicted sites. Then there is a phylogenetic network N' with the same number of galls as N , where each gall contains only conflicted sites.*

Proof: We consider a single gall Q , and examine the P -side of Q in detail. The argument for the S -side is symmetric. If the P -side only contains unconflicted sites, then these are removed using Lemma 2.3. So assume that the P -side contains both conflicted and unconflicted sites.

Let v be the last branching node on the P -side of Q . By Theorem 2.2, any sites below v must be unconflicted. There can be no sites in R_r below v , for any such site would have an all-zero column in M , a contradiction. Any sites in L_r below v have value 1 at the recombination node x

and can be moved to the edge out of x , as in the proof of Lemma 2.3. Hence, we assume that all sites on P are above node v , and so there is a branching edge below every site on the P -side of Q .

If there are no additional unconflicted sites on the P -side of Q , then the transformation is complete. Otherwise, let p be an unconflicted site in L_r on the P -side of Q . If there is any site q in S_R , then by Theorem 2.2 p and q would conflict, a contradiction. So, if there is an unconflicted site in L_r on the P -side, then there can be no site in S_R , and hence no site in S_L (recall that by definition, such sites are conflicted), and so there must not be any conflicted sites on the S -side of Q . But then by Lemma 2.3, all the unconflicted sites and branching edges can be moved off of Q , and we assume that has been done.

Now consider p^* , the highest unconflicted site in L_r on the P -side. There cannot be a conflicted site i in P_L above p^* , since by Theorem 2.2, i would have to be in conflict with a site $j \in P_R$ above i with a branching node between j and i . That would create the conditions for p^* and j to be in conflict, a contradiction. So p^* must be above all sites in P_L .

Similarly, if there is any site j in R_r above p^* , then there can be no branching node between it and p^* . This follows because there is a branching edge below p^* , so by Theorem 2.2, sites j and p^* would be in conflict, a contradiction. So there can be no branching nodes above p^* and no branching node (or sites) on the S -side of Q . Therefore, we can move p^* to the edge e entering the coalescent node of Q , creating a modified phylogenetic network N' that derives M and contains one fewer unconflicted sites.

If there is a branching node v on the P -side of Q with no sites above v on Q , then we further modify N' to prepare for additional site removals. Let $e = (w', w)$ be the edge into the coalescent node w . Expand e into two edges (w', w'') and (w'', w) by inserting a new node w'' on e , below any sites that are on e . Then disconnect any branching edges out of v and reconnect these edges to w'' .

Iterating as above, we can remove all unconflicted sites in L_r on the P -side of Q . We let Q' denote the resulting gall at this point, and now discuss how to handle any unconflicted sites in R_r on the P -side of Q' .

We make two observations. First, the only sites in L_r on the P -side of Q' are conflicted sites, i.e., in P_L . Second, every unconflicted site p in R_r must be below every site j in P_R . To see this, note that j must be in conflict with a site $i \in P_L$ below j on the P -side of Q' , so there must be a branching node between j and i , and a branching edge below i . Hence if p were above j , the conditions would be satisfied for p to conflict with i , a contradiction. So either all the unconflicted sites on the P -side of Q' are together at the bottom of the P -side of Q' , just above the last branching node v , or there is an unconflicted site p in R_r above a conflicted site i in P_L .

In the latter case, we transform Q' to conform to the former case, as follows. Assume that p is the first (topmost) unconflicted site in R_r on the P -side, and i is the last site in P_L . There cannot be a branching node between p and i , or else they would conflict. However, if p is not immediately above i , consider a site q between p and i . If q is in R_r , there would have to be a branching node between p and q or they would have identical columns in M . If q is in P_L , there would have to be a branching node between q and i or else they would have identical columns in M . Hence if there is an unconflicted site $p \in R_r$ above a site in $i \in P_L$, there can only be one such pair, and the two sites must be adjacent with no branching node between them. But then, we can flip the positions of p and i and still have a phylogenetic network that derives M . At that point, all the unconflicted sites in R_r are together at the bottom of the P -side of Q' , just above the last branching node v .

Let k be the site (if any) just above the first unconflicted site p in R_r on the P -side of Q' . If there is a node between k and p , call it v' , and otherwise create a node v' between the sites k and p . If there is no site k above p , then set v' to the coalescent node w of Q' . Then remove the edge into x from node v , and create an edge from v' to x containing no sites. The result is a phylogenetic network that derives M where gall Q has been transformed to a gall Q' that has only conflicted sites, and all other galls remain unchanged.

The theorem is proved by repeating this transformation for every gall containing unconflicted sites. ■

From the proof of the above theorem, we observe that the internal arrangement of any conflicted sites on Q is the same in N and N' , and all other details of N remain the same.

Corollary 2.2: *If there is a galled-tree for M , then there is a galled-tree where the number of recombinations is exactly the number of (non-trivial) connected components of the conflict graph, which is the minimum number of recombinations that any galled-tree can have.*

In the remainder of the paper, whenever we assume the existence of a galled-tree T for M , we assume, without stating it, that the galls of T only contain conflicted sites.

3 Arranging the Gall Q_c

The one-one correspondence between non-trivial connected components and galls in a galled-tree greatly simplifies the task of creating a galled tree for M . We can focus independently on each non-trivial connected component C of the conflict graph, to determine how the sites on that component are arrayed on the gall Q_c , and how to select the recombination point for Q_c . In this section, we show how to efficiently accomplish these tasks.

3.1 Selecting the Recombination Point r on Q_c

Lemma 3.1: *If there is a galled-tree for M , then every non-trivial connected component C of the conflict graph 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.*

Lemma 3.1 gives a necessary condition that can be used to prove that certain sets of sequences cannot be derived on a galled-tree. For example, see figure 5.

Proof All the sites on C must mutate on a single gall Q , and Q has only a single recombination point r . By Theorem 2.2a), $i < r \leq j$ for any conflicting pair i, j in C where $i < j$. Therefore, each edge in C connects one site whose index is below r and one site whose index is at or above r .

■

Definition 3.1 *Given the bipartite graph for C , we call the side containing the smaller sites the L_c (left) side, and the other side the R_c (right) side.*

It is easy to find the bipartition and select r : let p be the largest node (site) in C which is connected only to larger nodes, and let q be the smallest node in C which is connected only to

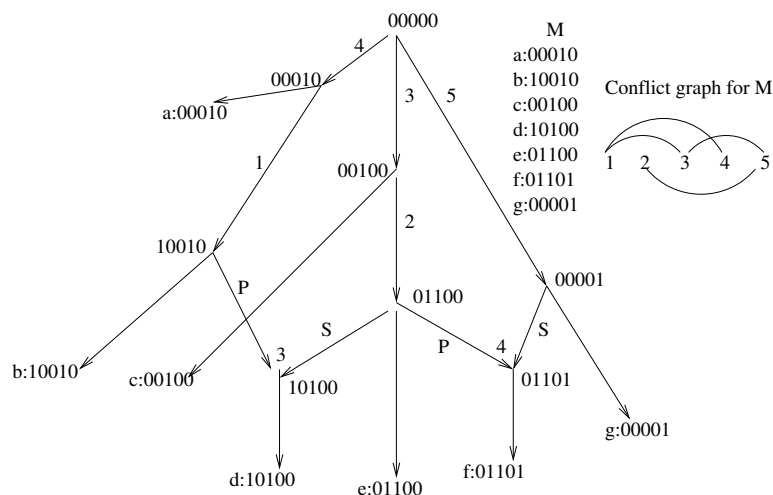


Figure 5: The phylogenetic network generates the sequences M on the right. Although the network is very similar to the one in figure 1, and only sequence g is different, M cannot be derived on a galled-tree. The conflict graph is bipartite and unique, but it does not have the required properties stated in Lemma 3.1.

smaller nodes. Then r can be chosen to be any integer strictly larger than p and less than or equal to q , and this defines L_c and R_c .

3.2 Arranging the Sites of C on Q_c

We now describe how to arrange the sites of C on a gall Q_c . Theorem 2.2 and corollary 2.1 will be the central tools. Let L and R denote the set of conflicted sites in a non-trivial connected component. Let P_L and S_L denote the sites in L on the prefix and suffix sides of a gall. Similar definition holds for P_R and S_R .

To begin, we can efficiently determine whether a pair of sites in L must be together on the same side of Q_C (even if we don't know which side that is), or must be on separate sides of Q_C .

Lemma 3.2: *Two pairs of sites i, i' in L must be on the same side of Q_C if and only if there is a row in M which has a 1 for both i and i' .*

Proof Suppose i and i' are on the same side of Q_C , and without loss of generality, suppose mutation i occurs before mutation i' . Consider the branch off Q_C just after the mutation for i' . Such a branch always exists; 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 1. 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_C , then at the recombination node x of Q_C , one parent node of x has (i, i') state-pair of (0,1) and the other parent has state-pair (1, 0). Since the recombination point does not occur between i and i' , the recombinant node x will have the (i, i') state pair equal to what it is at the parent of x on the P -side of Q_C . Therefore, the nodes on Q will only contain the (i, i') state-pairs of (0,0), (1,0) and (0, 1), and so by corollary 2.1, no row in M will have a 1 for both i and i' . ■

So, in $O(n)$ time we can determine if i and i' must be together on one side of Q_C or on separate sides of Q_C . 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_C , or partition L into two sets that must be on opposite sides of Q_C . So in $O(n|L|)$ time, we can uniquely partition the elements of L . Similarly, we can examine and uniquely partition the elements of R , in $O(n|R|)$ time.

In each component C , the time for the partition is $O(n|C|)$, and over all the connected components, the time is $O(n^2)$ since each site is in one connected component.

3.2.1 Initial Delineation of Conflicted Sites in Q_c

Definition 3.2 *For a non-trivial connected component C , we assume, as above, that L has been partitioned into two sets L_1 and L_2 (with L_2 possibly empty), such that all the sites in L_1 must be on one side of Q_C , and all the sites in L_2 must be on the other side. Similarly, R has been partitioned into two sets R_1 and R_2 (with R_2 possibly empty).*

Note that neither L_1 nor R_1 are empty, because in each conflicting pair, one column is in L and the other in R . We now analyze how information about the conflicted sites in C restricts the partition.

Definition 3.3 *For a site i , $\text{ONE}(i)$ is the number of 1's in column i .*

Lemma 3.3: *For a galled tree T for M , and a gall Q , suppose set P_L has at least two sites. The mutation for a site $i \in P_L$ must occur before the mutation for a site $i' \in P_L$ if $\text{ONE}(i) > \text{ONE}(i')$. Similarly, the mutation for a site $j \in P_R$ must occur before the mutation for a site $j' \in P_R$ if*

$\text{ONE}(j) > \text{ONE}(j')$.

Proof Let (v, v') be any edge branching off Q , where node v is not the recombination node x , and v is below the mutation i' . Then both i and i' have state 1 at v' , and by Corollary 2.1, every leaf below v' contributes 1 to both $\text{ONE}(i)$ and $\text{ONE}(i')$. But since i occurs before i' , and no two columns in M are identical, there must be a branching edge off Q between the mutation points for i and i' . Then, by Corollary 2.1 the leaves reachable from that branch contribute to $\text{ONE}(i)$ but not to $\text{ONE}(i')$. These statements also hold for $\text{ONE}(j)$ and $\text{ONE}(j')$. Now consider the recombination node x of Q , and note that the states of both i and i' are 1, and the states of both j and j' are zero. Hence every leaf below x which contribute to $\text{ONE}(i)$ also contributes to $\text{ONE}(i')$, and none of those leaves contribute to either $\text{ONE}(j)$ or $\text{ONE}(j')$. This proves the lemma. ■

Lemma 3.4: *The sites in P_L and P_R are interleaved on the P side of Q_C such that each site $i \in P_L$ is below every site in P_R that it conflicts with, and above every site in P_R that it does not conflict with. Also, the sites in S_L and S_R are interleaved on the S side such that each site $i \in S_L$ is above every site in S_R that it conflicts with, and below every site in S_R that it does not conflict with.*

Proof Follows directly by simple case analysis of Theorem 2.2. ■

Lemma 3.5: *Every site in P_L conflicts with every site in S_R .*

Proof According to Theorem 2.2, since there is a branch below the last conflicted sites of both the P and S sides of Q_C , for every site $i \in P_L$ and $j \in S_R$, we have rows that have values 01, 10 and 11 for the pair ij . ■

Definition 3.4 *Let $F(P_L)$ and $F(P_R)$ denote sets P_L and P_R ordered by the ONE counts, largest first. Let $f(P_L)$ and $l(P_L)$ be respectively the sites in P_L with the smallest and largest ONE values of all sites in P_L . Similar definitions hold for P_R and on the suffix side.*

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

We know that every site in P_R conflicts only with those in P_L and not with any of those in S_L . Similarly, S_L conflicts only with the sites in S_R . Also, every site in P_L conflicts with every site in

S_R . If any of these conditions are not met for the conflicted sites in a connected component C , then a phylogenetic network with recombination does not exist.

Let the total number of conflicts between R_1 and L_1 be A and that between R_1 and L_2 be B . If $A = 0$ and $B \geq 1$, then $R_1 = P_R$, $L_1 = S_L$, $L_2 = P_L$ and $R_2 = S_R$. Else, if $A \geq 1$ and $B = 0$, then $R_1 = P_R$, $L_2 = S_L$, $L_1 = P_L$ and $R_2 = S_R$. If both A and B are non zero, then we compute the total number of conflicts between R_2 and L_1 and R_2 and L_2 and make a similar analysis. If there are no set of sites in L and R with no conflicts between them, then we conclude that there is no solution. Otherwise, we proceed for a possible solution.

3.2.3 Case 2: $|L_2| = \emptyset$ and $|L_1|, |R_1|, |R_2| \geq 1$

Since P_L and not S_L conflicts with both P_R and S_R , and both R_1 and R_2 need to have sites with conflicts, $L_1 = P_L$.

Let every site in P_L conflict with every site in R_1 but not R_2 . Then, by Lemma 3.5, $R_1 = S_R$. A symmetric case exists if every site in P_L conflicts with every site in R_2 but not R_1 .

If every site in P_L does not conflict with every site of both R_1 and R_2 , then there is no gall that can represent the connected component C and hence a galled tree solution does not exist.

If every site in P_L conflicts with every site in both R_1 and R_2 , we can differentiate between P_R and S_R if $|P_L| \geq 2$ in the following way. If there is a sequence in which sites $(l(R_1), f(P_L), l(P_L))$ are $(1, 1, 0)$, then $R_1 = P_R$ and $R_2 = S_R$. Instead, if the corresponding similar sequence exists, $R_2 = P_R$ and $R_1 = S_R$. If neither sequence exists or if both the sequences exist, then there is no gall for C .

If every site in P_L conflicts with every site in both R_1 and R_2 , but $|P_L| = 1$, then R_1 and R_2 cannot be uniquely identified as P_R or S_R and can be represented interchangeably. Hence 2 different galls exist for C .

A symmetric analysis exists when $|R_2| = \emptyset$ and $|R_1|, L_1$ and $L_2 \geq 1$

3.2.4 Case 3: $|L_2| = |R_2| = \emptyset$ and $|L_1|, |R_1| \geq 1$.

There are 3 possible galls as listed below. Refer figure 4

Case W1: $L_1 = P_L$ and $R_1 = S_R$.

Case W2: $L_1 = P_L$ and $R_1 = P_R$ interleaved on P side.

Case W3: $L_1 = S_L$ and $R_1 = S_R$ interleaved on S side.

When $|L_1|$ and $|R_1| \geq 2$, we can identify the 3 cases as follows. Only case W2 is possible if and only if there is a sequence in which sites $(l(R_1), f(L_1), l(L_1))$ are $(1, 1, 0)$. Only case W3 is possible if and only if there is a sequence in which sites $(l(L_1), f(R_1), l(R_1))$ are $(1, 1, 0)$. Only case W1 is possible if both the above possibilities fail and if every site in L_1 conflicts with every site in R_1 . Note that the cases W2 and W3 account for both of the following possibilities.

- (1) every site in L_1 conflicts with every site in R_1
- (2) not every site in L_1 conflicts with every site in R_1

If $|L_1| = |R_1| = 1$, we cannot differentiate among the 3 cases based on leaves and they are all workable.

If $|L_1| = 1$ and $|R_1| \geq 2$, only case W3 is possible if there is a sequence in which sites $(f(L_1), f(R_1), l(R_1))$ are $(1, 1, 0)$. Otherwise, the cases W1 and W2 are workable.

If $|L_1| \geq 2$ and $|R_1| = 1$, only case W2 is possible if there is a sequence in which sites $(f(R_1), f(L_1), l(L_1))$ are $(1, 1, 0)$. Otherwise, the cases W1 and W3 are workable.

3.2.5 Interleaving Sites on a Side of Gall

After the sites in P_L, P_R, S_L and S_R have been identified, we need to group them on the prefix and suffix sides. We now examine how P_L and P_R can be interleaved on the prefix side of gall Q . A symmetric case exists for the suffix side.

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

In the case that at least one of P_L and P_R are non-empty, we determine a "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 previous placed site from $F(P_R)$, and immediately above the lowest k_j sites of P_L .

This creates an arrangement of P_L and P_R that is consistent with the conflicts P_L and P_R are involved with. To build the gall, for each site on a side of a gall, create an edge labeled with that site. Connect the edges serially from left to right and insert a node at the intersection of two sites. Then add a node and a dummy edge (without any site labels) to the last edge in that order. Finally, create a coalescent node w and connect it to the edges for f_p and f_s on each side. Also create a recombination node x and connect the dummy edges at the end of each side to x .

The gall thus formed is not yet complete. We need to specify other galls and trees of unconflicted sites that hang off of the different branching edges of Q . We will examine them later on.

Following are the observations on where nodes and branching edges occur on P .

Insert a node and a branching edge off of Q between any consecutive pair of sites s, t (where s occurs before t on Q) unless s is in L and t is in R . If s is in L and t is in R , then there must be a site u in L below t . Insert a node and a branching edge between s and t if and only if there is a sequence in M that has 1,0,0 for s, t, u . This will create a branching edge between every consecutive pair of sites s, t if and only if there is an s, t pair 1,0 in some sequence of M . Also, insert a node and a branching edge off of Q below the last site on P , but above the recombination node for Q .

After the algorithm builds a galled tree, we need not explicitly make the above checks for each gall. Instead, we can perform an overall test to verify if the generated galled tree matches the input data. If that is true, then the above conditions will automatically be satisfied due to the nature of the combinatorics involved.

We have thus observed that there are at most 3 different ways in which a gall can be arranged for any connected component C . Also, the arrangement of one gall is independent of another. Thus,

the total number of possible galled-trees for M is the product of the number of ways in which each gall can be arranged.

4 Connecting the Galls in a Galled-tree

Now we explain how to connect the galls together into a single galled-tree. Let T be a particular galled-tree for M and let Q and Q' be two galls in T . Gall Q is an "ancestor" of a gall Q' in T if there is a directed path in T for some node on Q to the coalescent node of Q' . If neither gall is an ancestor of each other, then we say that they are "incomparable". The algorithm to connect the galls will first deduce the ancestry relations between pairs of galls. We will see that the ancestry relations are invariant over all the galled-trees for M .

Since T is a particular galled-tree for M , the arrangement of sites on Q and Q' is determined. In that arrangement, let f_p and f_s be the first sites on the P and S sides respectively on Q . Define f'_p and f'_s similarly for Q' . Assume, without loss of generality, that f_s and f'_s exist. The analysis is symmetric for the other three combinations, one of which must exist. Also, let i, j be a pair of sites on Q that conflict with each other. Note that at the recombination node for Q , the state of at least one of i or j is set to 1, say i . Note that i might be f_p or f_s . Similarly, there is a site i' that has state 1 at the recombination node for Q' .

Now let Z' be any row of M with a 1 in column f'_s (there must be one since f'_s is involved in a conflict). If Q is an ancestor of Q' then Z' must have a 1 in at least one of the columns for f_s, f_p or i . Similarly, let Z be any row of M with a 1 in column f_s . If Q' is an ancestor of Q then Z must have a 1 in at least one of the columns for f'_s, f'_p or i' . So Q and Q' are incomparable if and only if neither of these conditions hold for rows Z' and Z .

After the initial preprocessing of M and collecting information in appropriate data structures, rows Z' and Z can be found in constant time, and in constant time we can check those (up to six) entries in rows Z' and Z . So in constant time we can determine whether Q and Q' are comparable or not. If comparable, then we have found a row which has a 1 in a column q for a site that appears on Q and a 1 in a column q' for a site that appears on Q' . That means that there is a path from

the root of T that passes through both the edges where sites q and q' mutate. We claim that Q is an ancestor of Q' if and only if the site q has strictly more 1's in its column than does site q' . To see this, note first that by Lemma 2.1, site q appears before q' on the path if and only if site q has a 1 in every row where site q' has a 1. Moreover, for any conflicted site on a gall, there must be at least two nodes on that gall where that site has state 1, so a tie for the largest number of 1's in columns q and q' is not possible. Hence,

Theorem 4.1 *In constant time, we can determine if Q and Q' are comparable, and if so, determine which is the ancestor of the other. There are at most $O(m) = O(n)$ galls, so over all the pairs of galls, we can determine the ancestry relations in $O(n^2)$ time.*

A gall Q is called the "immediate ancestor" of a gall Q' in T if Q is an ancestor of Q' and no descendant of Q is an ancestor of Q' . Every gall in T that has an ancestor in T , has an unique immediate ancestor, and the ancestor relation computed above is the transitive closure of the immediate ancestor relation. Hence, given a fixed arrangement of the sites on each gall, to find the immediate ancestor (if any) of each gall, we find the transitive-reduction of the ancestor relation. This can be done in $O(n^2)$ time because each gall has an unique immediate ancestor.

This identifies for each gall Q' , its immediate ancestor Q in T , or determines that Q' has no ancestor. Since we are ignoring unconflicted sites, every site appears on some gall, so in T a gall is connected to its immediate ancestor by a single edge (rather than a path). If Q is the immediate ancestor of Q' , we next want to determine the specific node, call it $v(Q, Q')$, on Q which is connected by a single edge to the coalescent node of Q' in T .

Let i, j be a conflicting pair on Q and let F be a sequence in M with a 1 for f'_s or f'_p . We claim that the 0/1 state of the i, j pair at recombination node x of Q is found at no other node on Q . Hence $v(Q, Q')$ is node x if and only if F has the same 0/1 state for i, j that is found at x . If that determination finds that $v(Q, Q')$ is not x , then F must have a 1 for exactly one of f'_p or f'_s , which identifies the side of Q that $v(Q, Q')$ is on. We then walk from the coalescent node of Q along that side until either encountering the edge e into the recombination node, or encountering the first edge e containing a site i such that F has state 0 for i . Node $v(Q, Q')$ is the node immediately

above the head of edge e . Since each of the $O(n)$ sites is on at most one gall, the total time to find all these nodes is $O(n)$.

We let T' denote the set of digraphs determined to this point, i.e., each consisting of all the arranged galls connected by the immediate ancestry edges. When there is more than one gall with no ancestor, we have a forest of galls. The different arrangements of sites on a gall merely permute the positions of the nodes and the branching edges attached to them. This clearly does not change the ancestry and immediate ancestry relations. Therefore, we can use any permitted arrangement of the sites on the galls to determine T' and

Theorem 4.2 *The set of digraphs T' is unique up to the different permitted arrangements of nodes inside the galls.*

Theorem 4.2 is a reflection of the "essential uniqueness" of the galled-trees that derive M .

Corollary 4.1 *Given the galls, and an arrangement of the sites on the galls, T' can be determined in $O(n)$ time.*

Further, if there is a galled-tree for M , any T' determined at this point can be extended to a galled-tree for M , by placing the unconflicted sites on edges of T' between galls, and possibly adding new edges containing unconflicted sites, or new edges leading to leaves.

5 Adding the Leaf Sequences and Unconflicted Sites

To finish constructing the galled-tree for M , we must add in the unconflicted sites, and place the sequences of M at the leaves, possibly adding additional tree edges outside of any galls. We now look at the construction in detail.

5.1 Constructing the Top Perfect Phylogeny in a Galled-tree

T' consists only of a forest of galls and accounts for all the conflicted sites in M . However, there could be some unconflicted sites in M in the galled-tree T that form a perfect phylogeny T_t near the root of T to which each of the gall forests are attached. To build T_t , we need the set of sequences S_t that define it. Note that in every sequence $K \in S_t$, all the conflicted sites are set to

0. This is true because there are no galls or parts of galls on the path from the root to K . With the appropriate data structures, we can find S_t in $O(mn^2)$ -time. Using the algorithm in [1], we then construct a perfect phylogeny in $O(mn)$ -time for the sequences in S_t . Thus, in $O(mn^2)$ -time, we can construct T_t . T_t now has nodes corresponding to site mutations and branching edges below those nodes with leaves corresponding to sequences in S_t .

Note that there could still be some unconflicted sites in T_t branching off of nodes in T_t that have not yet been discovered. This happens when there is no sequence in which the unconflicted site is 1 and all other conflicted sites are 0. They will be discovered when a tree of galls in T' is added to T_t as explained in the section below.

5.2 Connecting the Gall Forest to the Top Perfect Phylogeny

Every tree of galls in the gall forest T' must be connected to T_t . In particular, the coalescent node w of the 'root' gall in every tree of galls must be connected to a node in T_t with other possible edges for unconflicted sites between them. We detail how to add the 'root' gall of a particular tree of galls to T_t . This must be done repeatedly for every galled tree in the forest. We prove the following useful lemma.

Lemma 5.1 *Every site above the coalescent node of a gall has a greater ONE count than every site in the gall.*

Proof In a galled tree T , let the edge branching immediately below a site 'a' be connected to the coalescent node 'w' of a gall Q . Let $i \in L_r$ and $j \in R_r$ be two conflicting sites in Q . i and j can be arranged in one of the three possible ways as shown in figure 4. Suppose i and j are arranged as in W1. As per Theorem 2.2, there must be at least one sequence below each of i and j . All sequences below i have sites i and a set to 1 but with site j set to 0. Similarly, all sequences below j have sites i and a set to 1 but with site i set to 0. All sequences below the recombination node x contribute to the ONE count of all the three sites. Thus $\text{ONE}(a) > \text{ONE}(i)$ and $\text{ONE}(a) > \text{ONE}(j)$. But suppose i and j are arranged as in W2. All sequences below j have sites a and j set to 1 but with site i set to 0. All sequences below 'x' have sites a and i set to 1 but with site j set to 0. Thus

$\text{ONE}(\mathbf{a}) > \text{ONE}(\mathbf{i})$ and $\text{ONE}(\mathbf{a}) > \text{ONE}(\mathbf{j})$. Similarly, we can prove the same when the arrangement is as in *W3*. ■

Let Q be the 'root' gall of a tree of galls in the forest T' . Let f_p and f_s be the topmost sites in the prefix and suffix sides of Q . Let Z be a sequence in M which has the site f_p or f_s set to 1. Without loss of generality, let that site be f_p . Let U_z be the set of unconflicted sites in Z that are set to 1. If a galled-tree exists for M , then U_z is the set of all the unconflicted sites above Q plus one or more of the unconflicted sites hanging off of f_p . From U_z , delete all the sites with ONE counts less than or equal to $\text{ONE}(f_p)$. By lemma 5.1, U_z now has only the unconflicted sites above Q . Thus U_z is the list of all the sites from the root of T_t to w .

To find where to connect Q to T_t , starting from the root of T_t , walk down one of the edges whose site corresponds to a site in U_z . Continue walking down until there are no more outgoing edges or there is no edge whose site is in U_z . Let T_Q be the node immediately below the last edge traversed. Note that T_Q could also be the root of T_t . If all the sites in U_z are accounted for by the nodes from the root of T_t to T_Q , we simply connect 'w' to T_Q with a dummy edge (no site mutation) and we are done. Otherwise, for each unaccounted site in U_z , create a corresponding edge and a dummy node (corresponding to no sequence in M) immediately below it and connect all those edges and nodes in a linear fashion (like a singly linked list). Then, connect the edge at the head of this list of nodes to T_Q , add a dummy edge to the tail node of this list and connect it to 'w'. Note that all the new dummy nodes created for each of the unaccounted sites do not have a corresponding sequence in M because if there were such a sequence, the sites would have appeared in T_t when it was built. Conversely, we could also label all the unaccounted sites in U_z to a single edge.

After all the gall forests are connected, we now have a single digraph T_1 , (a galled-tree) of nodes (for the unconflicted sites near the root) and galls.

5.3 Constructing Subnetworks Hanging off of Galls

In addition to the unconflicted sites that formed the top perfect phylogeny near the root, there could be other unconflicted sites in M . These unconflicted sites form tree structures that hang off of either side of galls or below the recombination nodes; including cases when they can be suspended from an edge between two galls. To construct those tree structures, we need to collect the set of sequences that will occur immediately below each of the nodes in every gall. We accomplish that as follows.

First, for each sequence Z in M , we will find the node v_z in T_1 such that in any galled tree for M , v_z is the last node in T_1 on the path from the root to leaf Z . To do this, we do a bottom up traversal of T_1 , only traversing a gall after traversing all its descendants. At the start of the bottom up traversal, all sequences in M are unmarked. Then mark all the sequences that were used in the construction of the top perfect phylogeny near the root. Note that for each of those sequences, there is no node v_z . Let i, j be a conflicting pair that appears on a gall Q . We traverse Q as follows. Declare the recombination node x to be node v_z for every unmarked sequence which has the same i, j states as in x ; mark all of those sequences. Then traverse one side of Q , and for each node v (corresponding to site p) encountered, declare node v to be v_z for each unmarked sequence with a 1 in position p ; mark all of those sequences. Then traverse the other side of Q in a similar manner. The traversal takes $O(n^2)$ -time and finds the node v_z for each sequence Z in M . We now have the set of sequences immediately below each node in every gall.

To construct the tree structures hanging off of each node in a gall, we do a bottom up traversal of T_1 , constructing the trees for a gall only after constructing trees for all its descendant galls. Within a gall, we build the tree below the recombination node x and then go bottom up on each side of the gall building trees below each node encountered. We now describe how to build the tree below a particular node v . There are some differences in the procedure when v is a recombination node or on the side of a gall. We will explain those differences as we go along. Note that we do not collect sequences branching off the coalescent nodes and hence there are no trees hanging off of them.

Let S_v be the set of sequences established to be below node v . To construct a tree(perfect phylogeny) with v as the root, we need to reset every 'on' site at v to 0 and correspondingly reset to 0 all the sites in S_v that were reset in v . We claim that if there is a galled-tree for M , then, after the modification, the sequences in S_v must form a perfect phylogeny. If a perfect phylogeny cannot be built, we immediately conclude that there is no galled-tree for M . After building the perfect phylogeny T_S , we relabel the leaf sequences to the corresponding original sequences in M . Note that node v can have some galls connected to it as its children in T_1 . In the final galled-tree T , however, there could be some unconflicted sites on the path from v to each of its child galls. All those unconflicted sites are now part of the perfect phylogeny T_S . Therefore, we connect each of the child galls of v one-by-one to T_S (just as we connected the head gall in each gall forest to the top perfect phylogeny T_i ; so new unconflicted sites might also get added to T_S in the process) after disconnecting them from their parent galls in T_1 . Finally, we merge the 'root' of T_S so obtained with the node v . This is achieved by deleting any children that v may have and adding all the children of the 'root' of T_S as the children of v . Once this is done, we now have a probable galled-tree T_2 for M .

We now discuss in detail how to identify the set of 'on' sites in node v . Identify the set of sites O_v that are set to 1 in every sequence in S_v . This can be done in $O(mn)$ -time. Now O_v consists of all the 'on' sites in v and possibly other sites mutating below v without a branching node immediately below them. To delete those other sites, we make use of the easy observation(using Theorem 2.2) that a site i in a gall has a strictly greater ONE count than every site in the subnetwork hanging off of the node immediately below i . Let v be a node on the side of a gall and let k be the site labeling the edge immediately above v . Now, delete from O_v all the sites with ONE counts less than $\text{ONE}(k)$ and we are done. This can be done in $O(mn)$ -time. Instead, suppose v were the recombination node x . Then, choose site k to be one of P_L or S_R so that it is set to 1 at node x . Then $\text{ONE}(k)$ will again be greater than all the sites below x and we proceed in the same fashion. In this process, some other sites in P_L and S_R with lesser ONE counts than $\text{ONE}(k)$ could have been deleted. So, we add the remaining sites in P_L and S_R to O_v and remove the possible duplicates

from O_v . Thus we can find the set of 'on' sites at any node v in $O(mn)$ -time.

5.4 Handling Duplicate Sites

Suppose an input matrix M having duplicate sites satisfies the conditions of Lemma 3.1. Then, remove all but the first site in each set of duplicates and the resulting matrix be M' . If there exists a galled-tree $T_{M'}$ for M' , then there also exists a galled-tree T_M for M . To derive T_M from $T_{M'}$, we first relabel the sites in $T_{M'}$ to correspond to the sites in T_M and then add to each site labeling the edges, the set of its duplicate sites. Since the matrix M satisfied the conditions of Lemma 3.1 and because of the way in which the recombination point r is selected, all the duplicates of a site in a gall will have the same value at the recombination node. Thus, all the sequences generated in T_M will retain the same set of duplicate columns.

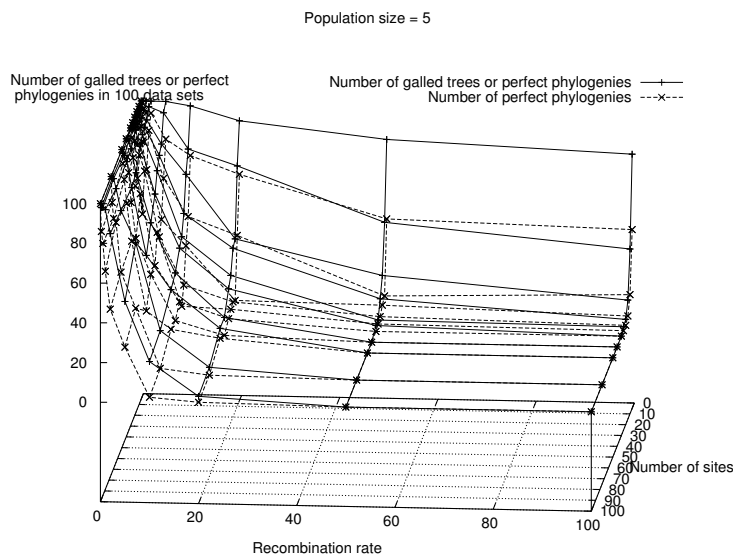
Conversely, if there exists a galled-tree T_M for M , there always exists a galled-tree $T_{M'}$ for M' . This is achieved by simply deleting the duplicate sites from T_M . Since all the duplicate sites mutate on the same edge in a galled-tree as the original site, we simply relabel the edges in T_M to exclude the duplicate sites and also delete the corresponding sites from the sequences generated.

6 The Final Check

Note that our algorithm works on the principle that if a galled-tree exists for a set of input sequences M , it will find it. We often make decisions based on examining a particular sequence in M assuming that a galled-tree exists. So, the network T_2 that is generated may or may not be a galled-tree for M . Thus, we perform the following two tests on T_2 .

1. Perform a top-down traversal on T_2 and compute the state of each site at every node. The alleged sequence in M branching out of every node in T_2 must satisfy the computed state of each site at that node.
2. Every site i must mutate at most once in T_2 .

If either of these conditions fail, we can immediately conclude that no galled-tree exists for M .



7 Results

In order to study the nature of occurrence of galled-trees in genomic data, we have implemented the above algorithm. The program is written in PERL and has been compiled for Linux. We generated genomic data using the generator cited in [6]. We were particularly interested in studying how the frequency of occurrence of galled-trees varied as a function of the number of sites, recombination rate and the number of individuals in a sample. We generated 100 sets of data for each setting and executed our program to test how many could have evolved as phylogenetic networks and how many among those were actually perfect phylogenies (no galls). The results are displayed graphically.

We found that galled-trees occur most frequently when the recombination rate and the number of sites are both low. The frequency decreased gradually as the number of sites were increased. The rate of decrease was steeper when the recombination rate was increased. Some of the recombinations are not visible in the generated data because there are no mutations along those sites. When there are a larger number of sites, there is a higher possibility of mutation and hence more of the recombinations become visible. Such visible recombinations get tested against the algorithm, resulting in higher failure rates for detecting the presence of galled-trees. When both parameters were increased sufficiently, there were no phylogenetic networks. As expected, when the recomb-

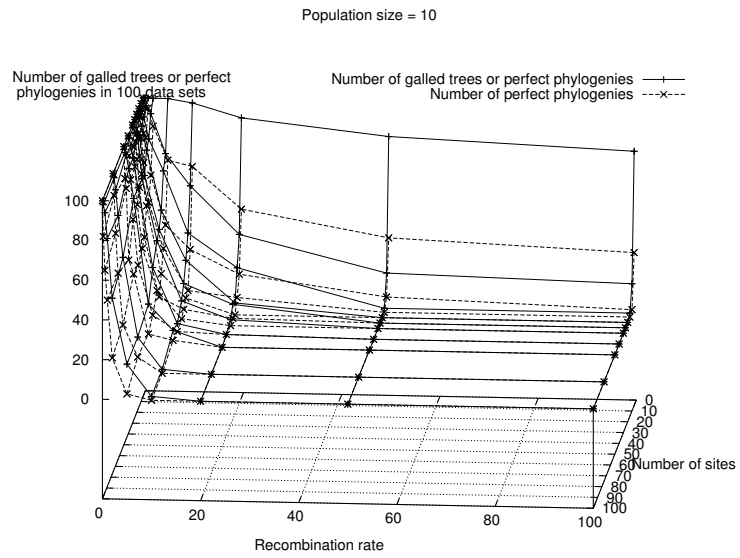


Figure 7: Population size = 10

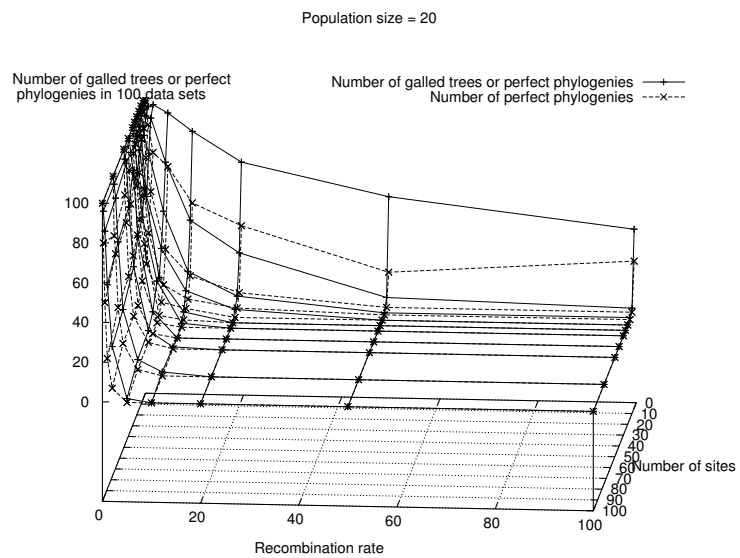


Figure 8: Population size = 20

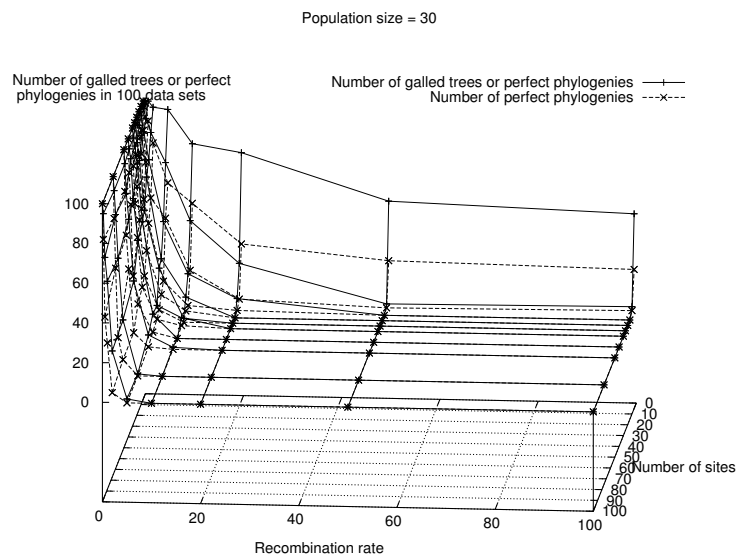


Figure 9: Population size = 30

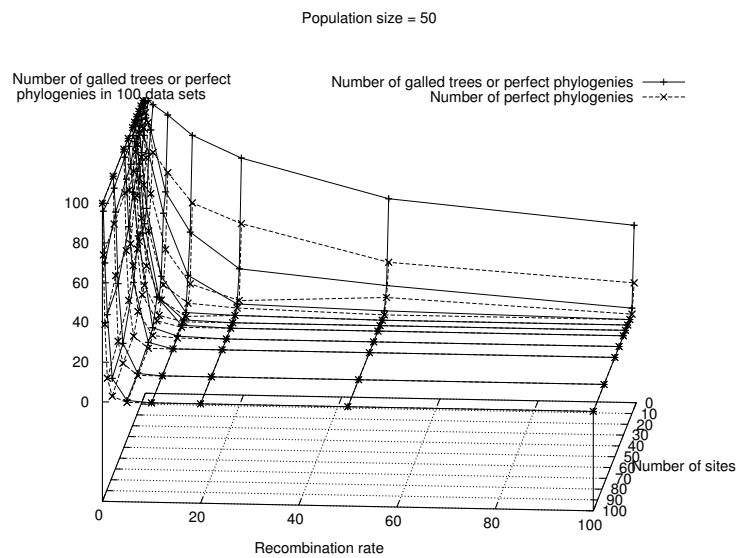


Figure 10: Population size = 50

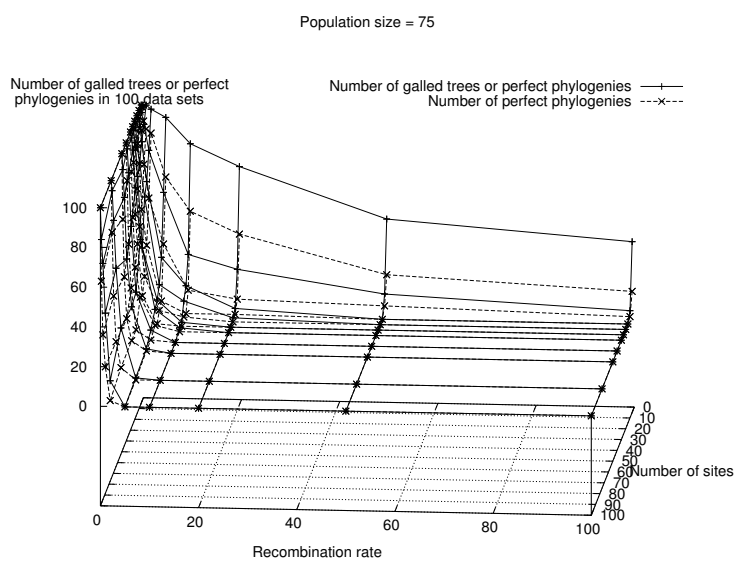


Figure 11: Population size = 75

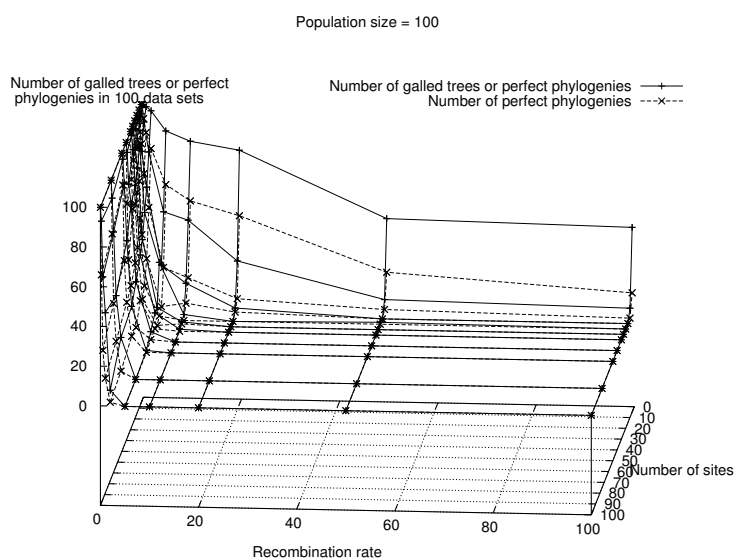


Figure 12: Population size = 100

nation rate was zero, all genomic data fitted into the perfect phylogeny model irrespective of other parameters. This change in the frequency of occurrence of phylogenetic networks as a function of both the parameters was more accentuated for larger population sizes.

As a curious experiment, we tried to find out the percentage of galled-trees that the algorithm in [10] failed to detect. In our simulations, we observed that the algorithm in [10] failed to detect as much as 5-65% of the galled-trees. The failure rates were very high for modest recombination rates, large number of sites and a large population size, indicating that most of the galls were laid out as per the patterns W2 and W3 in figure 4.

8 Relation to the Back-mutation Model

Another deviation from the perfect phylogeny model that is of interest is to allow a limited number of back-mutations, but no recombinations. A back-mutation is a mutation from state 1 back to state 0 that occurs on an edge, i.e., it is not a change due to recombination.

Theorem 7.1 *Any set of sequences M that can be derived by a galled-tree, can be derived by a true tree (no recombinations and hence no underlying undirected cycles) with at most one mutation and one back-mutation per site.*

Proof We take a galled-tree T for M and transform each gall Q separately, so that no cycles remain, but all the node labels are preserved. The simplest case is that Q has one side, say S , which has no mutations (sites). Remove the S -side (which consists of just a single edge into x) from Q . Let p denote the P -side parent of x . Then for any site i which has state 1 at p , but has state 0 at x , write a back-mutation for i on the (p, x) edge. Hence Q no longer is a cycle, but all the node labels on Q remain unchanged. The more complex case is that both the S and P sides have at least one mutation. In this case, remove the first edge on Q out of the coalescent node, on either the P or the S -side, say the S -side, and reverse the direction of all the remaining edges on the S -side. Next, for every site i that has state 1 at p but state 0 at x , write a back-mutation for i on the (p, x) edge. For every site i that has state 0 at p but state 1 at x , write the mutation i on edge (p, x) . Let s denote the parent of x on the S -side of Q . For every site i that has state 1 at s ,

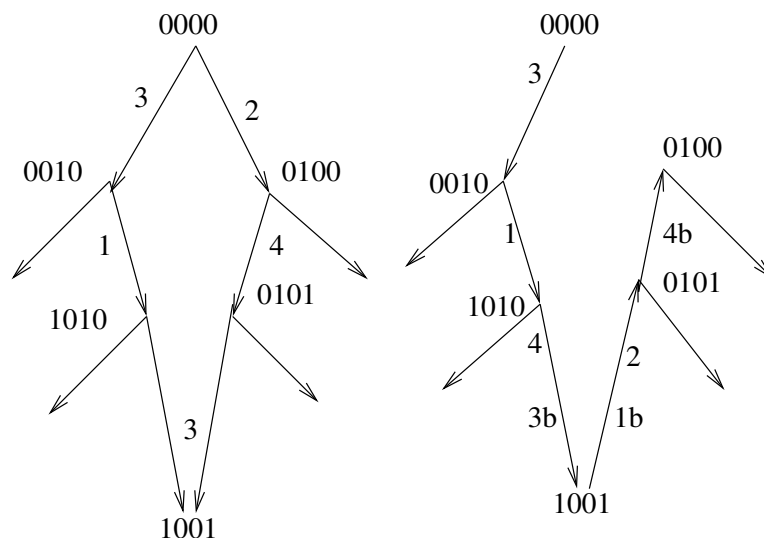


Figure 13: Gall Q is shown on the left and the result of the transformation is shown on the right. The recombination point for Q is 3, written above the recombination node. A number written on an edge is a mutation; a number followed by the letter b denotes a back-mutation.

but state 0 at x , write the mutation i on the (x, s) edge (which now runs from x to s). For every site i that has a state 1 at x , but state 0 at s , write the back-mutation for i on the (x, s) edge. Finally, convert each original mutation on the remaining edges of the S -side to a back mutation. The result is that Q is no longer a cycle, but all the node labels are preserved. Processing each gall in this way creates a true tree that derives M using at most one back-mutation per site. See figure 13 for an example. ■

9 One Provable Lower Bound on m_M

Hudson and Kaplan [7] and Myers and Griffith [9] give methods for computing lower-bounds on m_M . The methods in [9] seem very promising and may do well in practice, but what has been proved about lower bound methods is very limited. In particular, no existing (efficient) lower-bound method has ever been proved to have the property that it can always correctly determine if $m_M > 1$, i.e., if more than one recombination is needed. While this is a modest result, the algorithm in this paper does provably have that property. A phylogenetic network using just one recombination is a galled-tree, and so the conflict graph must consist of a single component. In

that case, the algorithm in this paper will construct a galled-tree with a single recombination. Conversely, if the algorithm cannot build a galled-tree for M , or cannot build one with just a single recombination, then m_M must be at least two.

10 Future Work and Open Questions

Future Work: The key idea introduced in this paper is the one-one correspondence of connected components of the conflict graph and galls in a galled-tree. More generally, properties of constrained phylogenetic networks more complex than galled-trees can also be elucidated through structural properties of the conflict graph. For instance, when a non-trivial connected component cannot be described by a gall, we can show that it can be described by a set of recombination cycles that share edges with each other. We call such a maximal subset of connected recombination cycles as a blob. We are presently developing that viewpoint and those results will be reported in a subsequent paper.

Open questions: There are many open questions. The most immediate are how to handle the case when the ancestral sequence is not known, how to handle missing data, and how to solve the perfect phylogeny haplotyping problem [2] when the underlying sequences (haplotypes) were derived on a galled-tree. We are also studying the optimality in terms of the number of recombinations in phylogenetic networks with blobs.

11 Acknowledgment

I am very thankful to Prof.Daniel Gusfield for inspiring me by his own intense love for the subject, for guiding me along every step on the path to my thesis, for being a patient, kind, tolerant and friendly advisor and for providing financial support when I needed it most. I thank Dr.Vladimir Filkov and Dr.Nina Amenta for kindly reviewing and approving my thesis proposal at a short notice and for co-operating with me in meeting my personal deadlines for thesis submission. I am very indebted to my beloved room mates who made many sacrifices to help me spend adequate

time in my academic work and research. Thanks are also due to my parents, sister, my many teachers of the past most of whom I have thanklessly forgotten, my friends at times of need and to many other millions of life forms from whom I have taken help directly or indirectly all my life. Finally, I thank that eternal, unseen guiding hand of a higher dimension that has made it possible for me to be what I am. I offer my humble respects to every one of you.

References

- [1] D.Gusfield. Efficient algorithms for inferring evolutionary history. *Networks*, 21:19-28, 1991.
- [2] D.Gusfield. Haplotyping as perfect phylogeny. Conceptual Framework and Efficient Solutions (Extended Abstract). In *Proceedings of RECOMB 2002: The Sixth Annual International Conference on Computational Biology*, pages 166-175, 2002.
- [3] D.Gusfield, S.Eddhu, and C.Langley. Efficient reconstruction of phylogenetic networks (of SNPs) with constrained recombination. In *Proceedings of 2'nd CSB Bioinformatics Conference*. IEEE Press, 2003.
- [4] J.Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Math. Biosci*, 98:185-200, 1990.
- [5] J.Hein. A heuristic method to reconstruct the history of sequences subject to recombination. *J. Mol. Evol.*, 36:396-405, 1993.
- [6] R.Hudson. Generating samples under a Wright-Fisher neutral model of genetic variation. *Oxford University Press*, 18:337-338, 2002
- [7] R.Hudson and N.Kaplan. Statistical properties of the number of recombination events in the history of a sample of DNA sequences. *Genetics*, 111:147-164, 1985.
- [8] J.D.Kececioglu and D.Gusfield. Reconstructing a history of recombinations from a set of sequences. *Discrete Applied Math.*, pages 239-260, 1998.

- [9] S.R.Myers and R.C.Griffiths. Bounds on the minimum number of recombination events in a sample history. *Genetics*, 163:375-394, 2003.
- [10] L. Wang, K. Zhang, and L.Zhang. Perfect phylogenetic networks with recombination. *Journal of Computational Biology*, 8:69-78, 2001.