UC Davis Computer Science Technical Report CSE-2004-31

Optimal, Efficient Reconstruction of Root-Unknown Phylogenetic Networks with Constrained and Structured Recombination[1]

Dan Gusfield

November 25, 2004

[1]To appear in JCSS Special Issue on Computational Biology.

Optimal, Efficient Reconstruction of Root-Unknown Phylogenetic Networks with Constrained and Structured Recombination

Dan Gusfield[†]

Keywords: Molecular Evolution, Phylogenetic Networks, Perfect Phylogeny, Ancestral Recombination Graph, Recombination, Gene-Conversion, SNP, Recurrent Mutation

Abstract

Phylogenetic networks are models of sequence evolution that go beyond trees, allowing biological operations that are not consistent with tree-like evolution. One of the most important of these biological operations is (single-crossover) recombination between two sequences. An established problem [14, 15, 28, 27, 29, 19, 17] is to find a phylogenetic network that derives an input set of sequences, minimizing the number of recombinations used. No efficient, general algorithm is known for that problem. An efficient algorithm does exist for the problem when the network is constrained to be a "galled-tree", and the ancestral sequence for the galled-tree is specified in advance [9, 11, 10]. However, the more biologically realistic case is that no ancestral sequence is known in advance, and the only previous algorithmic solution for that case takes exponential time.

In this paper we give an efficient solution to the galled-tree problem when no ancestral sequence is known in advance, and show that the solution produced has very strong global optimality properties. We also indicate how these results generalize to other complex biological phenomena such as gene-conversion, lateral gene transfer, hybrid speciation, and back and recurrent mutation.

1 Introduction to Phylogenetic Networks and Problems

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, cross-species hybrid speciation, gene conversion, and mobile genetic elements, there is greater need to understand the algorithmics and combinatorics of phylogenetic networks on which extant sequences were derived [24, 25]. Recombination is particularly important in deriving chimeric sequences in a population of individuals of the same species. Recombination in populations is the key element underlying techniques that are widely hoped to locate genes influencing genetic diseases.

[†]Department of Computer Science, 3051 Engineering II, University of California, One Shields Avenue, Davis, CA 95616. Email: gusfield@cs.ucdavis.edu Research Supported by NSF grant EIA-0220154. Thanks to C. Langley and S. Eddhu and D. Hickerson for helpful conversations on this topic.

Hein [14, 15, 28, 27] introduced the *phylogenetic network problem (with recombination)*: Construct a phylogenetic network that derives a given set of binary sequences, minimizing the number of recombinations used. No efficient, general algorithm is known for that problem, and it is claimed to be NP-hard [29]. The minimization criteria is motivated by the general utility of parsimony in biological problems, and because most evolutionary histories are thought to contain a small number of observable recombinations. The assumption that the sequences are binary is motivated today, in studies of populations (where the individuals are all from the same species), by the importance of SNP data. In SNP data, each site can take on at most two states (alleles) [4]. At the cross-species level, the assumption that the sequences are binary is motivated by the evolution of macroevolutionary traits, which are either present or absent in a species [6].

Wang et al. [29] focused on a special case of the phylogenetic network problem, where the ancestral sequence for the network is assumed to be *known in advance*, and where the phylogenetic network is required to be a "galled-tree" (defined below), where all recombinations involve only single-crossovers. Gusfield et al. [9, 11] gave a complete and efficient algorithm to determine if the input sequences can be derived on a galled-tree. Under those conditions, that paper solves the most general, special-case of the phylogenetic network problem that has a known efficient solution. However, the more biologically important case is that the ancestral sequence is *not* known in advance. Then, the problem is to find (if one exists) a sequence S, such that there is a galled-tree with ancestral sequence S that derives the input sequences. One can of course run the prior algorithm [9, 11] an exponential number of times, once for each possible choice of ancestral sequence, but that is impractical in general. Moreover, we would like to allow multiple-crossover recombinations. In this paper, we address the problem when no ancestral sequence is known, and when multiple-crossover recombinations are allowed.

1.1 Formal definition of a phylogenetic network

There are four components needed to specify a phylogenetic network that allows multiple-crossover recombination (see Figure 1).

A 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 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 some sequence R, called the "root" or the "ancestral" 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. The sequence labeling v is obtained from the sequence labeling v's parent by changing the state (from 0 to 1, or from 1 to 0) of the value at site i, for every integer i on edge e. This corresponds to a mutation at site i occurring on edge e.

b) For the recombination at node x, let Z and Z' denote the two m-length sequences labeling the parents of x. Then the "recombinant sequence" X labeling x can be any m-length sequence provided

that at every site i, the character in X is equal to the character at site i in (at least) one of Z or Z'.

The recombination "event" that creates X from Z and Z' is called a "multiple-crossover recombination". To fully specify the recombination event, we must specify for every position i whether the character in X "comes from" Z or Z'. This specification is forced when the characters in Z and Z' at position i are different. When they are the same, a choice of Z or Z' must be specified. For a given event, we say that a *crossover* occurs at position i if the characters at positions $i-1$ and i come from different parents. It is easy to determine the minimum number of crossovers needed to create X by a recombination of Z and Z'.

The sequences labeling the leaves of N are the extant sequences, i.e., the sequences that can be observed. We say that an (n, m)-phylogenetic network N *derives (or explains)* a set of n sequences M if and only if each sequence in M labels one of the leaves of N.

With these definitions, the classic "perfect phylogeny" [7] is a phylogenetic network without any recombinations. That is, each site mutates exactly once in the evolutionary history, and there is no recombination between sequences.

There are two restricted forms of recombination that are of particular biological interest. One is where X is formed from a *prefix* of one of its parent sequences (Z or Z') followed by a *suffix* of the other parent sequence. This is called "single-crossover recombination" since it uses exactly one crossover, and it is the definition of recombination used in [9, 11]. In that case, the parent sequence contributing the prefix can be denoted P, and the parent sequence contributing the suffix can be denoted S, and the labels P and S used in Figure 1 show which parent is the P-parent and which is the S-parent. The other case of restricted recombination is when X is formed from a prefix of one parent sequence, followed by an internal segment of the other parent sequence, followed by a suffix of the first parent sequence. This is a two-crossover recombination and when it occurs in meiosis, it is called "gene-conversion". In gene-conversion, the segment from the second parent is short, around 300 base pairs. It is believed [3] that during meiosis, single-crossover recombination is the dominant form of recombination occurring in intervals of DNA contained between neighboring genes, while gene-conversion is the dominant form of recombination in intervals of DNA contained inside a single gene. Gene conversion is known to be a very important molecular and genetic phenomenon, but it has been hard to study because of the lack of analytical tools and the lack of fine-scale data. In a different biological context, what we have defined as two-crossover recombination models the biological phenomena of "lateral gene-transfer" and "hybridization speciation".

What we have defined here as a phylogenetic network with single-crossover recombination is the digraph part of the stochastic process called an "ancestral recombination graph" in the population genetics literature (see [23] for example). 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. The term is more correct for a history of hybridizations, since those occur between species. However, in the Computer Science literature (and in some parts of Biology), the term "phylogeny" has come to be synonomous with "evolutionary tree", regardless of the actual 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 phylogenetic network problems studied in [17, 19, 14, 15, 28, 27, 11] all assume that recombination is single-crossover recombination. For continuity with those papers, we will first develop an

algorithm that produces networks that only use single-crossover recombination. However, the optimality of the networks produced by that algorithm will be proven in comparison to networks that allow multiple-crossover recombinations. In the last section of the paper, we indicate how to generalize the algorithm and the main result to allow multiple-crossover recombinations, and discuss the biological utility of such recombinations.

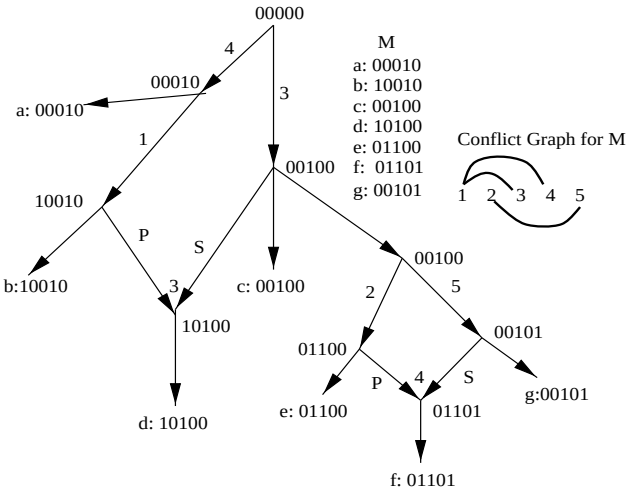

Figure 1. A phylogenetic network that derives the set of sequences M. The two recombinations shown are single-crossover recombinations, and the crossover point is written above the recombination node. In general the recombinant sequence exiting a recombination node may be on a path that reaches another recombination node, rather than going directly to a leaf. Also, in general, not every sequence labeling a node also labels a leaf.

1.2 Rooted and Root-Unknown problems

Hein's phylogenetic network problem is to construct a network that derives the input set of sequences, M, *minimizing* the number of single-crossover recombinations used. That problem can be addressed either in the rooted case, or the root-unknown case.

In the *rooted* phylogenetic network problem, a required root or ancestral sequence R for the network is specified in advance. In the *root-unknown* phylogenetic network problem, no ancestral sequence is specified in advance, and the problem is to select an ancestral sequence R, so that a phylogenetic network for M with ancestral sequence R minimizes the number of recombination nodes over all phylogenetic networks for M, and any choice of ancestral sequence.

Since no efficient general solution is known to Hein's problem, Wang et al. [29] introduced a biologically-motivated structural restriction of the problem.

1.3 A structural restriction

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. In Figure 1, the coalescent node of the top recombination cycle is labeled 00000 and the coalescent node of the bottom recombination cycle is labeled 00100. 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" or "mutates" or "is contained" on a gall Q if i labels one of the edges of Q. A phylogenetic network is called a "galled-tree" if every recombination cycle is a gall, and *only single-crossover* recombinations are allowed. The phylogenetic network in Figure 1 is a galled-tree.

If M cannot be derived on a perfect phylogeny, we would like to deviate from a tree by as little as necessary. Rather than having a phylogenetic network with a complex interleaving of cycles, it is preferable (if possible) to have a tree with a few extra edges, each creating a disjoint cycle. That is, we would like a galled-tree if possible, particularly if it does not use more recombinations than are used by more complex phylogenetic networks. Simulations have shown that when the recombination rate is low or moderate, galled-trees are frequently observed [5], particularly when no ancestral sequence is specified in advance. For example, when $n = m = 30$ and the recombination parameter r is set to 1 in Hudson's MS coalescent simulation program [16], about 85% of the datasets are derivable on a galled tree, and when r is set to 2, about 70% of the datasets have galled-trees. See [9, 11] for further motivation for galled-trees.

The **Galled-Tree Problem** is to determine whether or not an input set of sequences M can be derived on a galled-tree, when the ancestral sequence is known in advance. Wang et al. [29] introduced the problem, and provided an efficient algorithm that solves it in some, but not all, cases (see [11] for a discussion of this).

In [9] we developed an efficient algorithm ($O(nm + n^3)$-time) that solves the galled-tree problem. Further, we showed in [11], that when there is a galled-tree for input M, with a given ancestral sequence A, the algorithm creates a galled-tree that uses the minimum number of recombinations, over all phylogenetic networks for M *with* [1] *in [11] but was not always stated explicitly there.* This result holds even if multiple-crossover recombinations are allowed in the competing phylogenetic networks.

1.4 The Main Problem: The Ancestral Sequence is Usually Unknown

Usually, we do not know what the ancestral sequence is, and the existence of a galled-tree for M can depend on the specific ancestral sequence that is used. For example, there is no galled-tree with ancestral sequence 11000 for the matrix M shown in Figure 1, while, as shown, there is one when the ancestral sequence is 00000. Moreover, when there is more than one ancestral sequence which allows a galled-tree for M, the minimum number of recombinations needed (for specific ancestral sequences) can differ. For example, the sequences 01, 11, 10 can be derived on a perfect phylogeny without any recombinations, if any of those three sequences is the ancestral sequence, but if 00 is the ancestral sequence, then one recombination is needed. Hence we have the following two problems, which we address in this paper:

The Root-Unknown Galled-Tree Problem: Given M, find a sequence S such that there is a galled-tree for M with ancestral sequence S, or determine that there is none.

The Optimal Root-Unknown Galled-Tree Problem: If there is a galled-tree for M, find one that minimizes the number of recombinations over all galled-trees for M, and over all choices of the ancestral

[1] This last condition is implicit ancestral sequence A.

sequence. The solution is called an "optimal galled-tree" for M. The ancestral sequence of an optimal galled-tree for M is called an "optimal ancestral sequence" for M.

A secondary deficiency in the existing method is that we would also like to handle a broader range of biological phenomena than single-crossover recombination.

1.5 Main Results

We efficiently solve the Root-Unknown Galled-Tree Problem with an algorithm that also solves the Optimal Root-Unknown Galled-Tree Problem. The algorithm runs in $O(nm + n^3)$ time. We also establish the stronger result that an optimal galled-tree for M minimizes the number of recombinations over all phylogenetic networks for M (not just galled-trees) and all choices of ancestral sequence. This is true even if multiple-crossover recombinations are allowed in the competing network. The algorithm developed for single-crossover recombination can be extended to handle biological phenomena such as multiple-crossover, gene-conversion, lateral gene transfer and recurrent mutations.

2 Introduction to Tools and Solutions

The main tools that we use to solve the root-unknown galled-tree problem are two graphs representing "incompatibilities" and "conflicts" between sites. We introduce these graphs here.

Given a set of input sequences M, two columns i and j in M are said to be *incompatible* if and only if there are four rows in M where columns i and j contain all four of the ordered pairs 0,1; 1,0; 1,1; and 0,0. For example, in Figure 1 columns 1 and 3 of M are incompatible because of rows a, b, c, d. The test for the existence of all four pairs is called the "four-gamete test" in the population genetics literature.

Given a sequence S, two columns i and j in M are said to *conflict (relative to S)* if and only if columns i and j contain all three of the above four pairs that differ from the i, j pair in S [2].

Clearly, if a pair of columns i, j are incompatible, then i, j conflict relative to any sequence S. However, i, j may conflict relative to some sequence S, even though i, j are not incompatible. Finally, observe that if S is in M, then a pair of columns conflict relative to S if and only if they are incompatible.

2.1 Incompatibility and Conflict Graphs

We define the "incompatibility graph" $G(M)$ for M as a graph containing one node for each column (site) in M, and an edge connecting two nodes i and j if and only if columns i and j are incompatible. Similarly, given a sequence S, we define the "conflict graph" $G_S(M)$ for M (relative to S) as a graph containing one node for each column in M, and an edge connecting two nodes i and j if and only if columns i and j conflict relative to S. Figure 1 shows the conflict graph relative to the all-zero sequence S. This conflict graph is also the incompatibility graph for M.

A "connected component" (or "component" for short), C, of a graph is a maximal subgraph such that for any pair of nodes in C there is at least one path between those nodes in the subgraph. A "trivial" component has only one node, and no edges. The conflict graph in Figure 1 has two components. Let $cc_S(M)$ and $cc(M)$ be the number of non-trivial components in $G_S(M)$ and $G(M)$ respectively.

[2] In [9, 11], S was assumed to be the all-0 sequence, and the definition of conflict was specialized to that case, but the definition above is consistent with that earlier definition.

2.2 Prior structural results

The main structural result established in [9, 11] is

Theorem 2.1 *If T is any galled-tree for M with ancestral sequence A, then any gall in T that contains a site from a non-trivial component C of $G_A(M)$ contains all the sites of C, and contains no sites from any other non-trivial component. Further, any site from a trivial component that is on a gall Q can be moved to some edge touching Q, without otherwise changing Q.*

Clearly, if a gall has no sites, it can be contracted to a single node (if not eliminated entirely), so we assume that each gall contains some sites. We say that a galled-tree is a "reduced" galled-tree if no gall contains a site from a trivial component of $G_A(M)$. Every reduced gall has at least two sites, and at least three edges directed off of it. Theorem 2.1 establishes that if there is a galled-tree for M with ancestral sequence A, then there is a reduced galled-tree T for M with ancestral sequence A, and there is a one-one correspondence between the non-trivial components of $G_A(M)$ and the galls of T. The algorithm in [9, 11] produces a reduced galled-tree. Theorem 2.1 also leads to

Corollary 2.1 *Every reduced galled-tree for M, with ancestral sequence A, has exactly $cc_A(M)$ recombination nodes.*

It was also proved in [11] that if there is a galled-tree for M, then every phylogenetic network N for M with ancestral sequence A (N need not be a galled-tree) uses at least $cc_A(M)$ recombination nodes. This holds even if multiple-crossover recombinations are allowed in N. Hence, when there is a galled-tree for M with ancestral sequence A, we can efficiently solve Hein's phylogenetic network problem for M, (minimizing the number of recombinations), but only over networks that have the same ancestral sequence A.

The algorithmic consequence of Theorem 2.1 is that when trying to construct a galled-tree for M with known ancestral sequence A, we can focus on each non-trivial component C of $G_A(M)$ separately. Then, for each such component C of $G_A(M)$, we must determine how the sites can be arranged on a single gall; we must determine how the galls can be connected together in a tree structure; and we must determine where to place the sites from trivial components. All of these problems were solved efficiently in [9, 11], assuming an ancestral sequence A was *known*.

2.3 Moving to the Root-Unknown problem

We would like to follow the same approach for the root-unknown galled-tree problem. However, the algorithm in [9, 11] depends critically on knowning the ancestral sequence. The first, and main, difficulty now is that for any two sequences S and S', the conflict graphs $G_S(M)$ and $G_{S'}(M)$ may be different from each other. Hence without knowing which ancestral sequences allow a galled-tree (or if any will), we cannot even take the first step of the previous algorithm, i.e., building the appropriate conflict graph to identify its components. We focus next on that difficulty.

Theorem 2.2 *If there is a galled-tree for M with some ancestral sequence, then there is an optimal galled-tree for M where the (optimal) ancestral sequence is one of the sequences in M.*

Proof Let T be an optimal galled-tree for M, and let A be the ancestral sequence for T. By definition, every gall contains one recombination, and by the optimality of T, every gall Q in T must contain a pair of sites that conflict relative to A. Further, as established in [9, 11], every gall Q must have at least three edges branching off of it (i.e., each branching edge is directed from a node on Q to a node off of Q), and the fact that there must be at least two branches edges is even easier to establish. So, there is a least one edge branching off of Q from a node v which is not the recombination node of Q. It follows that there is a path P in T from the root to some leaf z which does not contain any recombination nodes. Let Z_z be the sequence labeling leaf z. Since T is a galled-tree for M, Z_z is in M.

Now consider rerooting T at node z, making Z_z the ancestral sequence, and reversing the directions of all edges on path P. Each such reversal of an edge e also changes the direction of the mutation on e, so for example if the original mutation had been from 0 to 1, it is now from 1 to 0. These reversals do not change any of the labels of nodes in T, nor do they change which node is the recombination node on any gall. Hence, the modified galled-tree, call it T', also derives M. The ancestral sequence of T' is Z_z, a member of M. Since T is optimal, and T' contains the same number of galls as T, T' is also optimal. □

Note that it is not true that every sequence in M can be used as an ancestral sequence of some galled-tree for M. We can completely characterize which sequences in M can serve as ancestral sequences, but omit that from this paper. It is also not true that if sequence Z in M is the ancestral sequence for some galled-tree for M, then Z is an optimal sequence. Still, Theorem 2.2 implies that both the Root-Unknown Galled-Tree Problem and the Optimal Root-Unknown Galled-Tree Problem can be solved in $O(n^2m + n^4)$ time by trying each sequence in M as the ancestral sequence, using the previous algorithm from [9, 11]. But, a faster algorithm, and more insightful result is possible, using the following

Theorem 2.3 *If there is a galled-tree for M, then there is an optimal (reduced) galled-tree for M with ancestral sequence A in M, where the graphs $G_A(M)$ and $G(M)$ are identical.*

This follows immediately from Theorem 2.2 and the fact that for any sequence S in M, $G_S(M)$ and $G(M)$ are identical.

Define $\mathcal{M}_M$ as the minimum number of recombination nodes used in any phylogenetic network deriving M, over any choice of ancestral sequence, even allowing multiple-crossover recombinations at any recombination node.

Theorem 2.4 *If there is a galled-tree for M, then $\mathcal{M}_M = cc(M)$, the number of non-trivial connected components of $G(M)$.*

Proof It was shown in [12, 1], that $cc(M)$ is a lower bound on $\mathcal{M}_M$. By Theorem 2.3, there is a reduced galled tree T with ancestral sequence A, where $cc_A(M) = cc(M)$, and by Corollary 2.1, the number of recombination nodes in T is exactly $cc_A(M)$. Hence, T is optimal and $\mathcal{M}_M = cc(M)$. □

Hence we can efficiently determine $\mathcal{M}_M$ even without knowing an optimal ancestral sequence, *if* there is a galled-tree for M.

3 Solving the Optimal Root-Unknown Galled-Tree Problem

In what follows, we assume that M can be derived on a galled-tree, and let T^* denote an arbitrary optimal galled-tree for M with optimal ancestral sequence A^*, where G_{A^*} and $G(M)$ are identical. By

Theorem 2.3, such a sequence A^* exists, and we can find the non-trivial components of $G_{A^*}(M)$ even though we don't know A^* or T^*. By Theorems 2.1 and 2.3, each gall in T^* contains all and only the sites of one non-trivial component C of $G(M)$, so we can also efficiently determine the sites that go on each gall in T^*. This is the first step in solving the Optimal Root-Unknown Galled-Tree Problem.

3.1 How to Connect the Galls of T^*

We next describe a method to determine how the galls are connected together in T^*, without knowing T^* or the internal arrangement of the sites on any gall. To do this, we first define a tree $\overline{T}$, (conceptually) created from T^*. For any sequence S and any set of sites on a component C of $G(M)$, define sequence $S(C)$ as the sequence S restricted to the sites in C.

We define $\overline{T}$ by conceptually transforming T^* into $\overline{T}$. Without loss of generality, we assume that every node v on any gall Q in T^* is incident with exactly one edge whose other end is off of Q. Such an edge is called an "off-edge", and might be directed into or out of v. To satisfy this assumption, we may need to make small, local modifications to T^*. For example, in the top gall in Figure 1, the node v labeled 00100 is incident with two edges whose other end is off that gall. To remedy this, we can simply create a new edge (v, w) from v to w, and then have two edges from w to the two endpoints of the two original edges out of v. If v is also the root of T^*, we create a new root node and connect it to v. The edge between them is the off-edge touching v. We also assume, without loss of generality, that each node on Q has a distinct sequence labeling it. We can always modify Q so that it has this property.

For any node v on a gall Q in T^*, let S_v denote the sequence labeling v, and let C_v denote the component in $G(M)$ whose sites are on Q. Label the single off-edge touching v in T^* with $(C_v, S_v(C_v))$. Note that if an edge connects two galls, then the edge will have two such labels. Those labels are in addition to any site (in a trivial component of $G(M)$) that might be on that edge in T^*. Finally, contract each gall Q in T^* to a single node q, label q with an identifier for the component C associated with Q, and make every edge undirected. The resulting undirected tree is $\overline{T}$. Figure 2 shows $\overline{T}$ derived from the tree in Figure 1.

Clearly, $\overline{T}$ specifies how the galls of T^* are connected to each other, although it does not show the internal arrangement of the sites on any gall, nor does it show where the root of T^* is. But, if we know $\overline{T}$, we know a substantial amount about T^*.

3.2 Constructing $\overline{T}$

We defined $\overline{T}$ (conceptually) from T^*, but algorithmically we will go in the other direction. We will construct $\overline{T}$ from M and $G(M)$, without knowning T^* or A^*. To do this, we use a classic theorem about tree reconstruction.

Let T be a tree where each leaf is labeled. The removal of any edge from T creates two connected subtrees, and partitions the leaves of T into two sets (each set is in one of the two subtrees). Each such bi-partition is called a "split", and each edge in T defines a distinct split. For a tree T, let $SP(T)$ be the family of all the splits, one for each edge. The classic *splits theorem* is:

Theorem 3.1 *The family of splits, $SP(T)$, uniquely determines tree T.*

There are many proofs of Theorem 3.1. One is obtained immediately from Theorem 3.1.4 (p. 44) in [26]. Also, if T has m leaves and n edges, then T can be uniquely reconstructed from $SP(T)$ in $O(nm)$ time [7, 26].

Given Theorem 3.1, the approach to constructing $\overline{T}$ is to learn $SP(\overline{T})$. The full explanation will involve reasoning about both $\overline{T}$ and T^*, so first observe that every edge in $\overline{T}$ is in T^*, and that every split defined by an edge e in $\overline{T}$ defines the same bi-partition of the leaves in T^*, when e is removed from T^*. Although the term "split" is only defined for a tree, we will also use it when referring to these bi-partitions in T^*.

The following theorem is the key observation about Galled-Trees that makes it possible to construct $SP(\overline{T})$, even though we don't know $\overline{T}$ or T^*.

Theorem 3.2 *Suppose an off-edge e in $\overline{T}$ is labeled with $(C_v, S_v(C_v))$. Then $S(C_v) = S_v(C_v)$ for every sequence S labeling a leaf of T^* on one side of the split of T^* defined by e, and $S'(C_v) \neq S_v(C_v)$ for every sequence S labeling a leaf on the other side of the split. Hence, the same is true for $\overline{T}$.*

Proof For notation, suppose e is the off-edge touching node q in $\overline{T}$, and q is derived from gall Q in T^*, which is associated with connected component C in $G(M)$. Node v is a node on Q, so we focus on Q and T^*. Let $e = (v, v')$ in T^*, and let A^* be the ancestral sequence in T^*.

We will prove the first part of the theorem for each node in Q, and then prove the second part of the theorem.

Consider the case that v is the coalescent node of Q in T^*, so e is directed from v' to v and $S_v(C_v) = A^*(C_v)$. That is, sequence S_v restricted to C_v is the same as the ancestral sequence A^* restricted to C_v. Moreover, since all the mutations for sites in C_v occur in Q, $S(C_v) = A^*(C_v)$ for any sequence S which is on the root side of the split in T^* defined by e. That same split occurs in $\overline{T}$, so $S(C_v) = S_v(C_v)$ for every sequence S labeling a leaf of T^* or $\overline{T}$ on the root side of the split defined by e.

Now consider the case that v is not the coalescent node of Q, so e is directed from v to v'. Clearly, $S_{v'}(C_v) = S_v(C_v)$ since no site in C_v can mutate on e. Let N' be the subgraph of T^* rooted at v'. Since all mutations of sites in C_v occur on edges in Q, no sites in C_v mutate in N'. Now let x be a recombination node in N' which is reached from v' without passing through any other recombination nodes. Restricted to the sites in C_v, the sequences labeling the two parents of x are identical, and so the recombination at x produces a a recombinant sequence which is identical to the parent sequences, when restricted to the sites in C_v. Hence, $S_x(C_v) = S_v(C_v)$. It follows then by induction on the number of recombinations encountered on the path from v, that $S_u(C_v) = S_v(C_v)$ for every node u in N', and in particular, for every leaf sequence S in N'. This proves the first part of the theorem.

To prove the second part of the theorem, note that for any site i *not* in C_v, the state of site i is the same at every node in Q. This is immediate for every node other than the recombination node x of Q, because site i does not mutate on Q. It remains true at x, because the state of i is the same in the sequence labels of both parents of x, so the recombination must retain the state of site i. It follows that, restricted to the sites not on C_v, all nodes on Q are labeled with the same sequence. Now if $S_v(C_v) = S_u(C_v)$ for two nodes v and u on Q, then $S_v = S_u$, which we assumed earlier could not happen. So every node on Q has a distinct sequence label, restricted to the sites in C_v. This fact, and the first part of the theorem, now establish the second part of the theorem. □

It is also true that if e is labeled with a site i from a trivial component of $G(M)$, then all the sequences on one side of the split have a value of 1 for site i, and all the other sequences have a 0 for site i.

Theorem 3.2 is important because it says that information about the node labels on a gall is reflected in the sequences at the leaves, and hence that information is contained in extant sequences.

3.2.1 Finding the splits family $SP(\overline{T})$

For each component C (trivial or non-trivial) in $G(M)$, define $M(C)$ to be matrix M restricted to the sites in C. The importance of Theorem 3.2 is two-fold. First each sequence $S_v(C)$ (which was defined relative to trees $\overline{T}$ and T^*, which we do not know), shows up as a sequence in $M(C)$, and each sequence in $M(C)$ is a sequence $S_v(C)$ for some node v on the gall associated with C. Second, the rows in $M(C)$ that contain sequence $S_v(C)$, identify exactly the split associated with the off-edge e touching v in $\overline{T}$.

We exploit this observation as follows: each distinct sequence Z in $M(C)$ defines a split in T^* and in $\overline{T}$: One side of the split defined by Z is the set of row indices in $M(C)$ whose rows contain sequence Z, and the other side of the split is the set of remaining row indices. When C is a trivial component, this approach defines the same split twice, but that causes no problem, and one of the copies can be deleted if desired. Hence, all of the splits of T^* and $\overline{T}$ that come from labeled edges in $\overline{T}$ can be efficiently found from M and $G(M)$. $\overline{T}$ may also so have unlabeled edges, but any unlabeled edge is incident with a leaf of $\overline{T}$, and hence correspond to splits with one site on one side and the remaining sites on the other side. We call these "leaf-splits". Recall, that a splits family from a tree uniquely defines the tree, and that the tree can be efficiently reconstructed from the splits family. So in summary,

Theorem 3.3 *Tree $\overline{T}$ (including its required edge labels) can be efficiently created from M and $G(M)$ by first creating a family of splits $SP(\overline{T})$ consisting of one split for each distinct sequence in $M(C)$, for each non-trivial connected component C of $G(M)$, and one or two identical splits for each trivial component of $G(M)$, and one leaf-split for each site in M.*

For example, with M from Figure 1, the two sets of restricted sequences $M(C_1)$ and $M(C_2)$, and the computed $\overline{T}$ are shown in Figure2. Since $\overline{T}$ is unique, we have the following

Theorem 3.4 *Tree $\overline{T}$ is invariant over all optimal galled-trees for M.*

3.3 From $\overline{T}$ back towards T^*

The next step in the solution of the Root-Unknown Galled-Tree Problem is to "re-inflate" the nodes in $\overline{T}$ that represent galls in T^*.

We first need to identify every node in $\overline{T}$ that was created (conceptually) by contracting a gall Q in T^* to a single node q. The key to this process is to note that each such node q in $\overline{T}$ is incident with more than one edge in $\overline{T}$, and that every edge incident with q has a label (C, Z), where the identifier C is the *same* on each edge incident with q. Further, this is true for no other nodes in $\overline{T}$. Using that fact, we can constructively and efficiently identify those nodes in $\overline{T}$ that must be expanded to become a gall. Moreover, from C, we know which sites are on the gall, and we know $M(C)$, the set of C-restricted node labels on the gall.

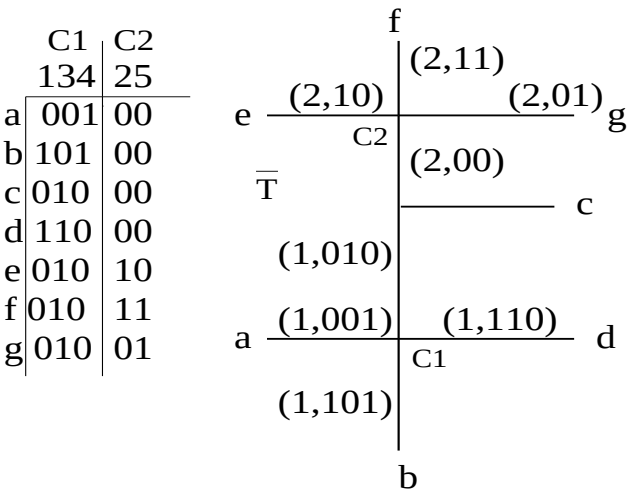

Figure 2. The sites of M are partitioned into two components of $M(G)$, and each is used to create a set of splits, along with the leaf-splits defined by each leaf (row in M). The tree $\overline{T}$ is the unique tree defined by these splits.

3.3.1 Arranging the sites of C on Q

We now describe how to arrange the sites of C on a gall Q. Since there may be some variability in how the sites can be arranged, We won't be able to reconstruct the original galled-tree T^* for sure, but we will still reconstruct an optimal galled-tree for M from $\overline{T}$. (Note however, it was shown in [11] that the variability is very small).

The method to arrange the sites on Q is a small modification of the method described in [9, 11] for the Rooted Galled-Tree Problem, and is specialized to the case that only single-crossover recombinations are allowed. Given C it is easy to determine a recombination point r that could be used on the gall Q containing the sites of C. See [9, 11] for more details.

To understand the method for arranging the sites, (conceptually) focus on a given arrangement of sites of C on gall Q in T^*, in isolation of the rest of T^*. Now remove the recombination node x from Q, remove the two edges entering x, and make the edges undirected. The resulting graph consists of a single path containing all the sites in C. Let u and y denote the ends of this path. For each node v on this path, add an edge from v branching off the path, and label its leaf end with $S_v(C)$. The result is an undirected perfect phylogeny, denoted $T(C)$, that by definition derives the sequences labeling the leaves of $T(C)$. Further, $S_x(C)$ can be formed by a single-crossover recombination of the sequences $S_u(C)$ and $S_y(C)$.

It follows from Theorem 3.2, that the leaf labels of $T(C)$ are exactly the sequences in $M(C)$, other than the sequence $S_x(C)$. That is, $T(C)$ is an undirected perfect phylogeny for all the sequences in $M(C)$ other than $S_x(C)$. Hence, we have

Theorem 3.5 *If there is a galled-tree for M, then there is a sequence X in $M(C)$, such that after removal of all copies of X, there is an undirected perfect phylogeny for the resulting matrix; the labeled edges of that perfect phylogeny contain all sites in C organized into one path; and a single-crossover recombination of the two "end" sequences $S_u(C)$ and $S_y(C)$ creates sequence X.*

It is a classic theorem that if a set of sequences can be derived on an undirected perfect phylogeny, then that perfect phylogeny is unique (except for the order of sites that are on the same edge). See [7, 8] for one exposition. Hence, given M and C, if we could guess the sequence X, we could create the unique undirected perfect phylogeny, which then would indicate a way to arrange the sites on Q. However, if we remove all copies of a different sequence Y, and yet there is an undirected perfect phylogeny for the resulting matrix, where all the sites in C are contained in one path, and the recombination of the two end sequences creates Y, then this other perfect phylogeny can also be used to arrange the sites on Q. We summarize these observation in the following algorithm.

Site-Arrangement Algorithm for gall Q corresponding to component C

1) Let $M(C)$ be matrix M restricted to the sites in C.

2) For each distinct sequence X in $M(C)$ do:

3) Let $M(C, X)$ be $M(C)$ after the removal of all rows with sequence X. Check if there is an undirected perfect phylogeny $T(C)$ for $M(C, X)$, where all sites on C are contained in one path whose end sequences can be recombined (with a single-crossover) to create sequence X.

If the answer is "yes", then output the pair $(X, T(C))$.

The first part of Step 3) is implemented by using the algorithm in [7] or [8] that tests if $M(C, X)$ can be derived by a perfect phylogeny $T(C)$. If $T(C)$ has two end points, labeled by sequences $S_u(C)$ and $S_y(C)$, we can test if X can be created by a recombination of $S_u(C)$ and $S_y(C)$ as follows: Find the length of the longest prefix of $S_u(C)$ that matches a prefix of X, and find the length of the longest suffix of $S_u(X)$ that matches a suffix of X. Let p_u and s_u denote these two lengths. Similarly, find p_y and s_y, which are defined for sequences $S_y(C)$ and $S_y(C)$. Then if X has length n, X can be obtained by recombining $S_u(C)$ and $S_y(C)$ if and only if $p_u(C) + s_y(C) \geq n$ or $p_y(C) + s_u(C) \geq n$.

Since we assumed there is a galled-tree for M, the Site-Arrangement Algorithm will find and output at least one pair $(X, T(C))$. However, a pair $(X, T(C))$ does not fully specify the arrangement of gall Q, because the choice of coalescent node has not been made. But $(X, T(C))$ does define the parents of the recombination node on Q, and hence does define the recombination node x, and also defines the circular order of the sites on Q. It is easy to see that *any* node v on Q, other than the recombination node x, can act as the coalescent node for Q with that circular arrangement: simply direct the edges on Q to form two disjoint directed paths from v to x. At that point, Q is fully specified. The particular choice of coalescent node will be made at a later point in the algorithm.

3.4 Choosing the arrangements and the root node

For each node q in $\overline{T}$ that represents a gall Q in T^*, we must replace q with Q, and arrange the sites on Q using one of the pairs $(X, T(C))$ found by the Site-Arrangement Algorithm, and we must choose a coalescent node for the arrangement. Of course, we must be careful to connect the nodes on Q to the correct edges: any node v on Q whose C-restricted node label is $S_v(C)$ must be connected to the (unique) edge incident with q that has label $(C, S_v(C))$ in $\overline{T}$. We must also choose a root for the galled-tree. However, the choices for the arrangements of the galls, and the choice for the root placement are not independent; one choice can constrain the others. The problem is that all edges in the final galled-tree T^* must be directed away from the root, and no edge can be directed into a recombination node of a gall. Since the arrangement of a gall specifies the recombination node, these choices are not independent. The arrangements of the galls (when there is a choice) must be coordinated with the choice of the root node.

We solve the coordination problem by directing some edges in $\overline{T}$, as follows. Suppose that for node q representing a gall Q in $\overline{T}$, the Site-Arrangement Algorithm finds *only one* pair $(X, T(C))$. Then in $\overline{T}$, we direct the (unique) edge in $\overline{T}$ that is labeled (C, X) away from q. This records the information that there is only one recombination node possible for Q, and the off-edge incident with that recombination node must be directed out of that node. After directing all such edges, any node v in $\overline{T}$ can be chosen as the root of $\overline{T}$, if and only if every node in (the partially directed) $\overline{T}$ can be reached from v using a path that does not go opposite to the direction of any directed edge. So the algorithm must find all such permitted points on $\overline{T}$ or declare that there are none.

After picking a permitted root point (if there is one), direct all of the edges in $\overline{T}$ away from the root. Then each node in $\overline{T}$ will have at most one edge directed into it, and if a node q has one directed edge into it and is expanded to a gall Q, the incoming edge defines the unique coalescent node w of Q. Finally, for any gall Q (associated with connected component C in $G(M)$), choose any $(X, T(C))$ found for C by the Site-Arrangement Algorithm, where the recombination node is not w. Add the node x, labeled with sequence X, to $T(C)$ and directed two edges into x from the two end-nodes of $T(C)$. That recombination cycle, along with the choice of coalescent node fully specifies the arrangement of gall Q, and doing this for every gall completes the construction of T^*.

3.5 Correctness and Time Complexity

Each step of the algorithm has been proven correct on the assumption that there *is* a galled-tree for M. The galled-tree created has exactly cc_M galls and recombination nodes, and hence is optimal using the lower bound mentioned earlier that $\mathcal{M}_M \geq cc_M$. If there is no galled-tree for M, then either some step of the algorithm will not be executable as described, or the algorithm will terminate but the graph produced will not be a galled-tree for M. So, algorithmically, one can simply check the output to see whether it is a galled-tree deriving M. If it is not, then there is no galled-tree for M. However, closer examination of the algorithm shows that when all steps complete, the graph produced is an optimal galled-tree. If there is no galled-tree for M, one of the steps of the algorithm will not be executable as described, and the algorithm will correctly conclude that there is no galled-tree for M.

For an n by m input matrix M, all of the steps of the algorithm can be implemented in $O(nm + n^3)$ time. The first $O(nm)$ term is for a radix sort of the columns of M to group together identical columns. It was established in [29] that when there is a galled-tree for M, the number of edges in it, and the number of distinct columns in M can be at most twice the number of distinct rows. So after removal of identical copies, the number of columns is $O(n)$. The $O(n^3)$ term is for finding the $O(n^2)$ incompatible pairs in M and building the graph $G(M)$. That worst-case bound can be reduced in theory because it is known [13, 2] how to to find all the pairs in the time needed to multiply two n by n matrices. The number of splits defined by M is $O(n)$ because $\overline{T}$ has $O(n)$ edges. $\overline{T}$ can be constructed in $O(n^2)$ time [7, 8], from the n by n matrix describing the splits. Given $M(C, X)$, the unique perfect phylogeny T for $M(C, X)$ (if there is one) can be found in $O(n^2)$ time by the same algorithm [7, 8], and testing if X can be formed by the recombination of the two end sequences of T can be done in $O(n)$ time. All the remaining steps take $O(n^2)$ time. Hence

Theorem 3.6 *Given an n by m input matrix M, an optimal galled-tree for M (if there is a galled-tree for M) can be found in $O(nm + n^3)$ time.*

The algorithm has been implemented in a Perl program galledtree.pl, which is available at

wwwcsif.cs.ucdavis.edu/~gusfield/galledtree.tar.

4 Extensions to other complex biological phenomena and structured recombination

So far, recombination in a galled-tree was assumed to be single-crossover recombination, and the solution to the Optimal Root-Unknown Galled-Tree Problem was developed only for single-crossover recombination. This was done for continuity with earlier papers. However, the algorithm is easily extended to allow multiple-crossover recombination at any recombination node, and multiple-crossover recombination can be used to model many complex biological phenomena. When multiple-crossovers are allowed at recombination nodes, but all recombination cycles are disjoint, we call the resulting network a "multiple-crossover galled-tree".

To modify the algorithm, we simply change Step 3) of the Site-Arrangement Algorithm as follows:

> 3) Let $M(C, X)$ be $M(C)$ after the removal of all rows with sequence X. Check if there is an undirected perfect phylogeny $T(C)$ for $M(C, X)$, where all sites on C are contained in one path whose end sequences can be recombined (allowing *multiple-crossover recombination*) to create sequence X.

Let $S_u(C)$ and $S_y(C)$ denote the two end sequences. We can test if X can be created by a multiple-crossover recombination of $S_u(C)$ and $S_y(C)$, starting with a prefix of $S_u(C)$, as follows:

> Set i to 1, and set Z to $S_u(C)$.
>
> Until (i is greater than the length of X) {
>
> Find the longest substring of Z starting at position i that matches a substring of X starting at position i. If there is none, then stop, and return "No". Otherwise, set i to one position past the right end of those matching substrings. If Z is $S_u(C)$, set Z to $S_y(C)$, else set Z to $S_u(C)$.
>
> }
>
> Return "Yes".

We can similarly test if X can be created by a multiple-crossover event, starting with a prefix of $S_y(C)$, and hence test if X can be created by a recombination of $S_u(C)$ and $S_y(C)$. If both tests return "Yes", then the one using the fewest number of crossovers also determines the minimum number of crossovers possible to create X from $S_u(C)$ and $S_y(C)$, and in some applications it may be desirable to use that one. The time for the modified Step 3) is clearly $O(n)$. For different biological applications, we can put a bound on the number of crossovers allowed.

Clearly, when the modified algorithm produces a multiple-crossover galled-tree for M, the number of recombination nodes used is cc_M, which is $\mathcal{M}_M$ by the lower bound result mentioned earlier. So the algorithm produces a phylogenetic network that is optimal with respect to the number of recombination nodes (or events) that occur. It is not necessarily optimal with respect to the number of crossovers used.

Conversely, suppose there is a multiple-crossover galled-tree T for M with some ancestral sequence. Theorems 2.1, 2.2, 2.3, 2.4, 3.4, and 3.5 are the keys to proving the correctness and optimality of the

solution to the Root-Unknown Optimal Galled-Tree Problem, when only single-crossovers are allowed. Each of those theorems is easily modified to extend to the case when multiple-crossover recombinations are allowed. We leave the details to the reader. In summary,

Theorem 4.1 *If there is a multiple-crossover galled-tree for M, then the modified algorithm will find one, and it will use the minimum number of recombination nodes over all phylogenetic networks for M and all choices of ancestral sequence. The time bound for the algorithm remains $O(nm + n^3)$.*

The algorithm to find a multiple-crossover galled-tree for M, or to determine that there is none, has been implemented as the program multicross.pl and can be found at wwwcsif.cs.ucdavis.edu/~gusfield/galledtree.tar.

4.1 Multiple-crossovers model complex biological phenomena

We have previously mentioned that "gene conversion" [3] can be viewed as a multiple-crossover recombination with exactly two crossovers. Gene-conversion occurs during meiosis, and is observed in population data (i.e., sequences taken from individuals of the same species). Through very different biological mechanisms, and often at a different biological scale, "hybrid speciation" and "lateral gene-transfer" cause the movement of genetic material between two sequences (often between two species) [22, 21, 18]. However, mathematically (but not biologically) these phenomena look like what we have defined as multiple-crossover recombination. Hence, the algorithm to find multiple-crossover galled-trees can be used to derive a set of sequences believed to have been created by mutation and hybrid speciation or] lateral gene-transfer. These models also have application in areas outside of biology, such as in linguistics [20].

Multiple-crossover recombination can also be used to model "back-mutation" or "recurrent-mutation". Back-mutation occurs when the state of a site mutates back to its ancestral state. Recurrent-mutation occurs when the state of a site is permitted to mutate from its ancestral state more than once in an evolutionary history. Each such mutation can be modeled as a two-crossover recombination in a phylogenetic network. For example, a single back-mutation at site i in a sequence Z can be modeled by the two-crossover recombination of the ancestral sequence A and sequence Z, where the prefix and suffix come from Z, and only site i comes from A. If the number of back-mutations is small, then the "recombination cycles" created by this modeling of back-mutations may be disjoint. We can modify Step 3) of the Site-Arrangement Algorithm to only allow a recombinant sequence X to be derived from the end sequences of $T(C)$ by a single back-mutation (or perhaps several back-mutations at different sites, if that is meaningful). One can again prove that *if* there is a phylogenetic network with back-mutations where all the recombination cycles are disjoint, then the modified algorithm will in fact find one which minimizes the number of back-mutations over all evolutionary histories that allow back-mutations and all choices of ancestral sequence. Recurrent-mutations can also be handled in a similar way, and more generally, the algorithm can be modified to allow general recombination, back-mutation and recurrent-mutation in an evolutionary history.

Thus the algorithm for solving the Root-Unknown Optimal Galled-Tree Problem is actually a general framework for efficiently minimizing the number of deviations from the perfect phylogeny model, provided that there is an evolutionary history for the sequences where the "recombinations cycles" (used to model the deviations) are disjoint. Disjointness is likely to occur when the number of deviations from the perfect phylogeny model is modest. Thus, we have described in this paper a general algorithmic tool for studying complex evolutionary phenomena, when the number of nonperfect phylogenetic events is modest.

References

[1] V. Bafna and V. Bansal. The number of recombination events in a sample history: conflict graph and lower bounds. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1:78–90, 2004.

[2] V. Bafna, D. Gusfield, S. Hannenhalli, and S. Yooseph. A note on efficient computation of haplotypes via perfect phylogeny. *Journal of Computational Biology*, 11(5):858–866, 2004.

[3] A. Berry and A. Barbadilla. Gene conversion is a major determinant of genetic diversity at the DNA level. In R.S. Singh and C.B. Krimbas, editors, *Evolutionary Genetics: From Molecules to Morphology*, pages 102–123. Cambridge University Press, 1999.

[4] A. Chakravarti. It's raining SNP's, hallelujah? *Nature Genetics*, 19:216–217, 1998.

[5] S. Eddhu. Personal Communication.

[6] J. Felsenstein. *Inferring Phylogenies.* Sinauer, Sunderland, MA., 2004.

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

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

[9] 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*, Los Alamitos, CA, 2003. IEEE Press.

[10] D. Gusfield, S. Eddhu, and C. Langley. The fine structure of galls in phylogenetic networks. *INFORMS J. on Computing, special issue on Computational Biology*, 16:459–469, 2004.

[11] D. Gusfield, S. Eddhu, and C. Langley. Optimal, efficient reconstruction of phylogenetic networks with constrained recombination. *J. Bioinformatics and Computational Biology*, 2(1):173–213, 2004.

[12] D. Gusfield and D. Hickerson. A new lower bound on the number of needed recombination nodes in both unrooted and rooted phylogenetic networks. Report UCD-ECS-06. Technical report, University of California, Davis, 2004.

[13] E. Halperin and R. Karp. Perfect phylogeny and haplotype assignment. In *Proc. of RECOMB 2004: The 8'th Ann. International Conference Research in Computational Molecular Biology*, pages 10–19. ACM Press, 2004.

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

[15] J. Hein. A heuristic method to reconstruct the history of sequences subject to recombination. *J. Mol. Evol.*, 36:396–405, 1993.

[16] R. Hudson. Generating samples under the Wright-Fisher neutral model of genetic variation. *Bioinformatics*, 18(2):337–338, 2002.

[17] 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.

[18] B. Moret, L. Nakhleh, T. Warnow, C.R. Linder, A. Tholse, A. Padolina, J. Sun, and R. Timme. Phylogenetic networks: Modeling, reconstructibility, and accuracy. *IEEE/ACM Transactions on Computatational Biology and Bioinformatics*, pages 13–23, 2004.

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

[20] L. Nakhleh, D. Ringe, and T. Warnow. Perfect phylogenetic networks: A new methodology for reconstructing the evolutionary history of natural languages. preprint, April 14, 2002.

[21] L. Nakhleh, J. Sun, T. Warnow, C.R. Linder, B.M.E. Moret, and A. Tholse. Towards the development of computational tools for evaluating phylogenetic network reconstruction methods. In *Proc. of 8'th Pacific Symposium on Biocomputing (PSB 03), pages 315-326*, 2003.

[22] L. Nakhleh, T. Warnow, and C.R. Linder. Reconstructing reticulate evolution in species - theory and practice. In *Proc. of 8'th Annual International Conference on Computational Molecular Biology*, pages 337–346, 2004.

[23] M. Norborg and S. Tavare. Linkage disequilibrium: what history has to tell us. *Trends in Genetics*, 18:83–90, 2002.

[24] D. Posada and K. Crandall. Intraspecific gene genealogies: trees grafting into networks. *Trends in Ecology and Evolution*, 16:37–45, 2001.

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

[26] C. Semple and M. Steel. *Phylogenetics*. Oxford University Press, UK, 2003.

[27] Y. Song and J. Hein. On the minimum number of recombination events in the evolutionary history of DNA sequences. *Journal of Mathematical Biology*, 48:160–186, 2003.

[28] Y. Song and J. Hein. Parsimonious reconstruction of sequence evolution and haplotype blocks: Finding the minmimum number of recombination events. In *Proc. of 2003 Workshop on Algorithms in Bioinformatics*, Berlin, Germany, 2003. Springer-Verlag LNCS.

[29] L. Wang, K. Zhang, and L. Zhang. Perfect phylogenetic networks with recombination. *Journal of Computational Biology*, 8:69–78, 2001.