PhyloNet: Phylogenetic Networks Toolkit

User Documentation

The BioInformatics Group
Department of Computer Science
Rice University
http://bioinfo.cs.rice.edu

Development and maintenance of the PhyloNet software package is made possible through generous support from the Department of Energy (grant DE-FG02-06ER25734), the National Science Foundation (grant CCF-0622037), the George R. Brown School of Engineering (Roy E. Campbell Faculty Development Award), and the Department of Computer of Science at Rice University.

Copyright

Contents

1 Introduction **4**
1.1 Contributors 4
1.2 Contact Info 4

2 Installation **5**
2.1 Usage Instructions 5

3 Software Conventions **6**
3.1 Phylogenetic Tree Representation: The Newick Format 6
3.2 Phylogenetic Network Representation: The eNewick Format 7

4 Tool Reference **9**
4.1 `countcoal` 9
4.2 `lca` 9
4.3 `mast` 10
4.4 `rf` 10
4.5 `recomp` 11
4.6 `riatahgt` 12
4.7 `charnet` 14
4.8 `cmpnets` 15
4.9 `netpars` 16
4.9.1 Maximum parsimony of phylogenetic networks 17

Chapter 1

Introduction

PhyloNet is a collection of tools designed mainly for analyzing, reconstructing, and evaluating reticulate (or non-treelike) evolutionary relationships, generally known as *phylogenetic networks*. Various methods that we have developed make use of techniques and tools from the domain of phylogenetic trees, and hence the PhyloNet package includes several tools for phylogenetic tree analysis.

PhyloNet is released under the GNU General Public License. For the full license, see the file `GPL.txt` included with this distribution. Though the source code has not yet been posted, it is available by request.

1.1 Contributors

PhyloNet is designed, implemented, and maintained by the BioInformatics Group, which currently includes Professor Luay Nakhleh (`nakhleh@cs.rice.edu`) and Ph.D. students Derek Ruths (`druths@cs.rice.edu`) and Cuong Than (`cvthan@cs.rice.edu`). The group is affiliated with the Department of Computer Science at Rice University.

1.2 Contact Info

Feel free to contact us by mail, email, phone or fax:

The BioInformatics Group
c/o Luay Nakhleh
Department of Computer Science
Rice University
6100 Main Street
Houston, TX 77005
Email: `nakhleh@cs.rice.edu`
Phone: 713-348-3959
Fax: 713-348-5930

Chapter 2

Installation

System Requirements In order to run the PhyloNet toolkit, you must have Java 1.5.0 or later installed on your system. All references to the *java* command assume that Java 1.5 is being used.

Downloading `phylonet.jar` Acquire the current release of PhyloNet by downloading the most recent version of the PhyloNet JAR file. You will have a file named `phylonet_vX_Y.jar`, where X is the major version number and Y is the minor version number.

Installing the file Place this in the desired installation directory. The remainder of this document assumes that it is located in `$PHYLONET_PATH/jar`.

Installation is now complete. In order to run PhyloNet, you must execute the file `phylonet_vX_Y.jar`, as described in the next section.

2.1 Usage Instructions

All tools are run by executing the jar, `phylonet_vX_Y.jar`, and specifying the tool name and arguments.

Executing `phylonet_vX_Y.jar` The downloaded file, `phylonet_vX_Y.jar`, is an executable jar. On the command-line, type

```
java -jar $PHYLONET_PATH/jar/phylonet_vX_Y.jar -h
```

This will run the tool and print the help message. The help message will include a listing of the tools available for execution.

Running tool `<tool>` To run the specific tool, `<tool>`, type

```
java -jar $PHYLONET_PATH/jar/phylonet_vX_Y.jar <tool>
```

Since each tool requires different input data and parameters, it is helpful to review the help for a given tool before using it. To see the help information, type

```
java -jar $PHYLONET_PATH/jar/phylonet_vX_Y.jar <tool> -h
```

For more detailed help information see the Tool Reference section of this document.

Chapter 3

Software Conventions

3.1 Phylogenetic Tree Representation: The Newick Format

Many tools require trees as input or produce trees as output. All of which will be in Newick format [2]. In brief, Newick format uses nested parentheses to model trees. A single tree is always terminated by a semi-colon.

Name and Distance Properties Every node can have two properties: a name and a distance from its parent. The name of a node must be a string with using only alphanumeric characters. This distance can be any positive, real number. In Newick format, these properties are separated by a ":" character. Thus, the node "foo" that is 0.63 units away from its parent is written

```
foo:0.63
```

Either of these properties can be omitted. If the distance is omitted, the colon is not written.

Specifying Children Internal nodes have other nodes as their children. This is specified by placing a parenthetical clause before the name and distance pair:

```
(...)foo:0.63
```

Within the parentheses, the node's children are written, separated by commas. The listing does not specify order:

```
(childA:0.1, childB, :0.3)foo:0.63
```

This specifies that the node "foo" has three children, one of which is unnamed and one of which has no distance specified.

Roots By default, a tree is rooted. Therefore, the tree:

```
(A,(B,C)I1)ROOT;
```

is rooted at the node named "ROOT". To override this and specify an unrooted tree, the entire tree specification must be preceded by the string `[&U]`. Therefore the following tree is not rooted:

```
[&U] (A,(B,C)I1)D;
```

3.2 Phylogenetic Network Representation: The eNewick Format

The Newick format for representing and storing phylogenetic trees was adopted in 1986 [2], and it has been the standard for almost all phylogeny software packages ever since. This format captures an elegant correspondence between leaf-labeled trees and matched parentheses, where the the leaves are represented by their names and the internal nodes by a matched pair of parenthesis that contains a list of the Newick representation of all its children. Shown in Figure 3.2 are three trees along with their representations in the Newick format.

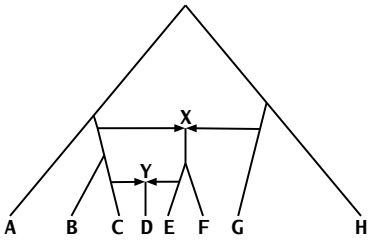
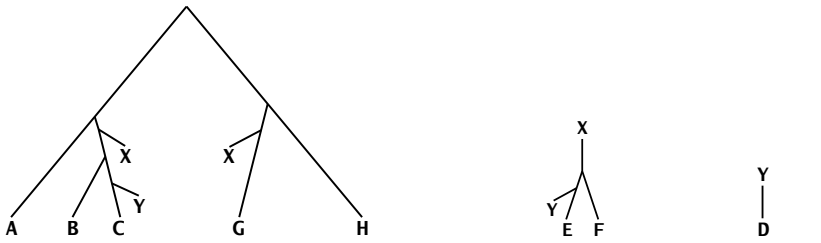


Figure 3.1: A phylogenetic network N with eight leaves (labeled $A, \ldots, H$) and two network nodes X and Y. Shown are the orientation of the network edges; all other edges are directed away from the root (toward the leaves).



$N' = ((A, (X, (B, (C, Y)))), ((X, G), H))$ $X = (F, (E, Y))$ $Y = (D)$

Figure 3.2: Three trees, N', X, and Y, along with their Newick representation. These three trees form the tree decomposition $\mathcal{F}$ of the phylogenetic networks N in Figure 3.1. The eNewick representation of N is the triplet $\langle N'; X; Y \rangle$.

However, no similar format exists for phylogenetic networks; instead, existing phylogenetic network software tools store these networks as adjacency lists of their underlying graphs, which are usually very large and necessitate translation of representations among the different tools. We propose a new format, which we call the *extended Newick*, or eNewick, format. In this format, a phylogenetic network is decomposed into trees, each of which is represented using the Newick format. Figure 3.2 shows the tree decomposition and eNewick representation of the network N in Figure 3.1.

The eNewick representation of phylogenetic network is the format used in the phylogenetic network characterization and comparison utilities in PhyloNet (described below).

Phylogenetic Network Representation: The Edge-list Format

The RIATA-HGT method outputs phylogenetic networks in the *edge-list* format. In the case of horizontal gene transfer, a species tree is almost always assumed, and hence the phylogenetic network can be represented by a pair $\langle ST, X \rangle$, where ST is the species tree in Newick format (with names assigned to internal nodes), and X is the list of edges posited between the edges of ST.

Chapter 4

Tool Reference

Descriptions of the tools available in the PhyloNet package are provided here. Tools are listed in alphabetical order.

General Usage When running a tool, the tool name and arguments must be preceded by:

```
java -jar $PHYLONET_PATH/jar/phylonet_vX_Y.jar
```

4.1 `countcoal`

Description This `countcoal` tool computes the number of coalescent scenarios that can explain the incongruence observed between two trees. The tool reads two trees from standard in (one tree per line) and prints the number of coalescent scenarios to standard out. The trees must be specified in newick format.

Usage

```
countcoal [-h] [-f ⟨infile⟩] [-o ⟨outfile⟩]
```

The default behavior described above can be modified by using the following options:

`-h` prints the help message

`-f` ⟨`infile`⟩ reads the pair of trees from the file ⟨infile⟩. This file should contain the trees on the first two lines—one tree per line.

`-o` ⟨`outfile`⟩ writes the output to the file ⟨outfile⟩ rather than the standard out.

Citing If you use this tool in published work, please cite [18].

4.2 `lca`

Description This `lca` tool computes the least common ancestor of a group of nodes in a tree. The tool reads the tree, T, as input first. The tool prints this tree with all internal nodes labeled. If a node is not labeled in the original tree, then the node receives a generated name. Then the tool reads sets of nodes. A node set is a collection of names

belonging to nodes in T. Syntactically, a set of nodes is a set of space delimited names on the same line. For each node set read, `lca` prints their least common ancestor.

Usage

`lca [-h] [-o` *filename*`] [-t` *filename*`] [-n` *filename*`]`

By default, the tree and node set are read from standard in and the results are written to standard out. This behavior can be modified by using the following options:

`-h` prints the help message

`-o filename` writes all the output to the file *filename*

`-t filename` reads the tree from the file *filename*

`-n filename` reads the node set from the file *filename*

4.3 `mast`

Description The `mast` tool computes a Maximum Agreement Subtree of a pair of trees, using the algorithm of Steel and Warnow [17]. The trees must be either all rooted or all unrooted. If the trees are rooted, then the rooted MAST is computed. If the trees are all unrooted, then the unrooted MAST is computed. All trees must also have at least three leaves.

Usage

`mast [-h] [-a] [-i` *filename*`] [-o` *filename*`]`

By default, the trees are read from standard in and the MAST is written to standard out. This behavior can be modified by the following options:

`-h` prints the help message

`-o filename` writes all the output to the file *filename*

`-i filename` reads all the input trees from the file *filename*

To compute *all* MASTs, specify an option `-a` in the command line arguments. However, since the number of MASTs may be exponential in the number of leaves in the trees, using this option can slow down the execution time (in some cases considerably). Plans for modifying the code so that it computes MASTs of more than two trees are currently under way.

4.4 `rf`

Description This tool computes the symmetric difference, also known as the Robinson-Foulds (RF) distance [14], between two trees. The trees do not need to be rooted. The first tree read is the *model* tree. The second is the *experimental* tree.

Usage

`rf [-h] [-m` *filename*`] [-e` *filename*`] [-o` *filename*`]`

The tool reads in two trees. It outputs the number of False Negative edges, the number of False Positive edges, the number of internal edges in the model tree and finally the number of internal edges in the experimental tree. By default, the tool reads the trees from the standard in and print the results to standard out. This behavior can be modified by the following options:

`-h` prints the help message

`-m filename` indicates that the model tree be read from file *filename*

`-e filename` indicates that the experimental tree be read from the file*filename*

`-o filename` writes all the output to the file *filename*

4.5 recomp

Description This tool is an implementation of the RECOMP algorithm for detecting recombination breakpoints along a set of aligned genetic sequences [15, 16]. For a detailed description of the algorithm, see [16]. In brief, the algorithm reads in a set of aligned genetic sequences, a window size, a step size, and a window comparison function. The function is designed in such a way that highly different adjacent windows correlate to windows separated by a recombination breakpoint. The method returns the value of the comparison function evaluated at positions along the length of the sequences.

Usage

`recomp [-h] -w` *WIN_SIZE* `-s` *STEP_SIZE* `[-d RF|SPR] [-i` *IFILE* `[`*FMT*`]] [-o` *OFILE*`] -c` *FXN PAUP_PATH POP_SIZE NUM_ITERS NUM_LVLS*

`-h` prints the help message

`-w WIN_SIZE` specifies the window size

`-s STEP_SIZE` specifies the step size

`-d RF|SPR` specifies what distance measure should be used. The Robinson-Foulds (*RF*) distance [14] is the default. The Subtree-Prune-Reroot (SPR) distance is also supported, and in this case the RIATA-HGT heuristic [10] is used to compute the distance.

`-i` *FILE FMT* specifies the file that should be used as input. If omitted, then input is read from the STDIN. By default, input is read as a fasta file. FMT can be used to override the default. *FASTA* indicates that the input is in FASTA format. *PLAIN* indicates that the input is in a plain format in which each line of the file has a sequence name and the sequence itself. The plain format also supports an optional header declaring the number of sequences and the number of nucleotides in the sequences.

`-o OFILE` specifies the file that will be used for output. If omitted, then STDOUT is used.

`-c` *FXN PAUP_PATH POP_SIZE NUM_ITERS NUM_LVLS* specifies the window comparison function that will be used.

- *PAUP_PATH* is the path to a PAUP executable
- *POP_SIZE* is a parameter used by PAUP. It specifies the number of trees that PAUP will maintain in memory during maximum parsimony search. The recommended value for this is 100.
- *NUM_ITERS* is a parameter used by PAUP, specifying the number of iterations to use during the maximum parsimony search. Values between 25 and 100 tend to yield accurate trees.
- *NUM_LVLS* specifies the number of tree levels that should be included in the window comparison. In [16], this is the k parameter. Taking more levels increases the number of non-optimal trees included in the window analysis. In [16], 3 levels were found to give the best result.

For an explanation of the different functions, please see [16]. Valid choices of *FXN* are:

- *PARS* - the parsimony difference function
- *MAX* - the average maximum distance
- *MIN* - the average minimum distance
- *INT* - the percent intersection of the window tree sets

Citing If you use this tool in published work, please cite [16].

4.6 `riatahgt`

Description This tool is an implementation of the RIATA-HGT algorithm for detecting and reconstructing horizontal gene transfer events from phylogenetic incongruence. For a detailed description of the algorithm, see [10]. In brief, the algorithm reads in a single species tree and any number of gene trees. It returns the horizontal gene transfer events.

Usage

`riatahgt [-h] [-u] [-p` *prefix*`] [-i` *filename*`] [-o` *filename*`]`

The tool reads a set of trees, either from the standard input or from the input file specified with the `-i` option. For the tool to run, there has to be at least two trees in the input. If it reads k trees ($k \geq 2$), it assumes that the first tree is the species tree and the remaining trees are the gene trees. The tool then proceeds by computing solutions on pairs of species/gene trees. In other words, if the input contains k trees $T_1, \ldots, T_k$, the tool computes solutions for the pairs (T_1, T_2), (T_1, T_3), $\ldots$, (T_1, T_k). Further, the tool prints the solutions for these pairs in this particular order.

For each pair, the tool first prints the species/gene trees, in this order, with the internal nodes labeled (this labeling is needed for printing the HGT edges). The user can

specify a particular prefix to name the internal nodes with the `-p` option. For example, if the user species `-p Int`, and assuming there are ℓ internal nodes, then the tool will label the internal nodes of the species tree as $\texttt{Int}_1$, $\texttt{Int}_2$, ..., $\texttt{Int}_\ell$. The internal node names in the HGT events refer to the names in the species, and not the gene, tree. Because solutions might share common HGT events, we group and print them by components to make the tool's ouput more concise and informative. From the tool's output, one can get a complete solution by selecting an event from each component. For example, let's consider the following species and gene trees:

```
ST = ((e,(f,g)I1)I2,((a,(b,c)I3)I4,d)I0)I5;
GT = (((a, b), c), (d, ((e, f), g)));
```

HGT events for this pair (`ST`, `GT`) are computed and printed by the tool as:

```
There are 3 component(s), which account for 27 solutions, each of size 3
----------------------------------------------------------------------------
Component I5:
Solution1:
        I5 -> I4 [time violation?]
Solution2:
        d -> I2
Solution3:
        I2 -> d
----------------------------------------------------------------------------
        Component I2:
        Solution1:
                f -> e
        Solution2:
                I2 -> g [time violation?]
        Solution3:
                e -> f
----------------------------------------------------------------------------
        Component I4:
        Solution1:
                I4 -> c [time violation?]
        Solution2:
                a -> b
        Solution3:
                b -> a
----------------------------------------------------------------------------
```

There are 27 possible solutions for this pair of trees, each of which consists of events from sub-solutions of each component. The set {`d -> I2, f -> e, a -> b`} is a solution, for example. Note that an HGT event has the format:

`sn` $\rightarrow$ `tn`

where *sn* and *tn* are the names of nodes in the species tree, and are the source and target, respectively, of an HGT edge. This indicates that an edge should be added from *sn* to *tn* in the species tree.

By default the all trees are read from standard in and the HGT events are written to standard out. This behavior can be changed by using the following options:

`-h` prints the help message

`-i filename` indicates that the species tree and gene tree(s) are read from file *filename*

`-o filename` writes all the output to the file *filename*

`-p prefix` specifies a name prefix used to label internal nodes. If no prefix is specified, then the default prefix `I` is used.

Input trees are always refined and contracted before RIATA-HGT computes HGT events. In general, this detects fewer HGT events than when the trees are left intact. In some cases, however, computing HGT events with the refined and contracted trees might produce more events than with the original trees (such cases are rare, though). To prevent the trees from being refined and contracted, use the option `-u`; RIATA-HGT will compute events for the original trees provided.

Citing If you use this tool in published work, please cite [10].

4.7 `charnet`

Description This tool implements functions to compute the trees, tripartitions and clusters contained in a phylogenetic network [8, 13, 12]. Please note that the clusters and tripartitions returned by this tool will not contain trivial ones, that is, clusters and tripartitions for the network's root and leaves.

Usage

```
charnet [-h] [-i input] [-o output] -m tree|tri|cluster
```

By default, the tool reads the input network from the standard input, and outputs the result to the screen. The options `-i` and `-o` allow the user to specify the files that store the input network and the result.

The user must include the option `-m`, followed by either `tree`, `tri` or `cluster`, to specify which characterization he or she wants. For example, if the user types

`charnet -i net -m cluster`,

then the tool will read the network from the file `net`, and compute all the clusters contained in this network. The result is printed on the standard output.

The option `-h` prints the help message on the usage of the tool. The network must be in the eNewick format.

4.8 `cmpnets`

Description This tool allows the user to compute the distance between two phylogenetic networks, based on their topologies. Three measures are currently implemented:

- Tree-based measure.
- Tripartition-based measure.
- Cluster-based measure.

See [11, 8, 13, 12] for complete description of these measures.

Usage

```
cmpnets [-h] [-i input1 input2] [-o output] -m tree|tri|cluster
```

By default, the tool reads the two networks from the standard input and prints the result to the screen. By default, it is assumed that the network is described using the eNewick format. However, the user has the option of describing the network as either a set of clusters or a set of trees. If the network is described as a set of clusters, then the first line of the network description contains a single word—"textttcluster", followed by the clusters, each listed on a separate line. The elements of a cluster are separated by white space. Similarly, if the network is described as a set of trees, then the first line of the network description contains a single word—"`tree`", followed by trees, each listed on a separate line using the Newick format.

For the two input networks, the only combinations of formats allowed are:

1. Both networks are in eNewick format. In this case, any of the three measures can be invoked.
2. Both networks are described in terms of their constituent trees. In this case, only the tree-based measure can be invoked.
3. Both networks are described in terms of their constituent clusters. In this case, only the cluster-based measure can be invoked.
4. One network is in eNewick format, and the other is described in terms of its constituent trees. In this case, only the tree-based measure can be invoked.
5. One network is in eNewick format, and the other is described in terms of its constituent clusters. In this case, only the cluster-based measure can be invoked.

When at least one of the two networks is not in eNewick format, and since in this case the type of measure used is automatically determined, entering the measure type is optional (actually, it will be ignored by the tool).

If the options `-i` and `-o` are used, the networks will be read from the two specified input files and the result is stored in the output file. In case the two networks are read from the standard input, they must be separated by a blank line. After the inputs are successfully parsed and the networks are constructed, the tool computes their distance. For example, if the user types

`cmpnets -o result -m tree`,

then the tool will wait for the user to enter two networks as input, compute the tree-based distance and then write the result to the file `result`. If the user already has clusters for a network, he/she can type

```
cmpnets -i net1 net2,
```

where `net1` contains a list of clusters for a network and a line ``cluster'' at the beginning of the file, and `net2` contains eNewick representation for another network. The tool will compute the cluster-based distance between the two networks, and prints the result to the screen. As in the tool `charnet`, input clusters for a network will not contain trivial clusters, i.e., clusters that have either one taxon or all the taxa in the network, since these clusters are present in all networks.

The option `-h` prints the help message on the usage of the tool.

4.9 `netpars`

Description This tools computes the parsimony score of a phylogenetic network, given the sequences at its leaves, using the formulation of [9] (reviewed briefly below). The sequences are split into blocks of equal length (with the exception of the last block, which many not necessarily contain the same number of sites as the ones before) which is specified by the user. The parsimony score is computed based on these blocks as well as the trees contained inside the network.

`Usage` To compute the parsimony, type:

```
netpars [-h] [-i netfile seqfile] -b block-size [-o output]
```

By default, the tool will read the network and sequences from the standard input. In this case, the input for the network and the sequences must be separated by a blank line. The tool can also accept input from files, if the option `-i` is enabled. In this case, you specify two file names, one for the file containing the network and the other for the sequences, as arguments for the option `-i`. The score will be printed on the standard ouput, but it can be directed to a file if the option `-o` is enabled.

The tool assumes that the input for networks is in the eNewick format. Refer to the subsection 3.2 for the specification of the eNewick format. For the file that contains the sequences, the tool expects the following format:

```
#taxa sequence-length
taxon-name1 sequence1
taxon-name2 sequence2
...
```

The first line in the file specifies the number of taxa and the length of every sequence. The number of taxa should match the number of leaves in the network. Each subsequent line contains a taxon name followed by the DNA sequence of this taxon. Note that all sequences must have the same length; otherwise, the tool will throw an I/O exception.

The tool currently does not handle weighted parsimony, a feature that will be added in the near future.

4.9.1 Maximum parsimony of phylogenetic networks

This relationship between a phylogenetic network and its constituent trees is the basis for the MP extension to phylogenetic networks described. We now briefly review the definitions of [9].

Definition 1 *The Hamming distance between two equal-length sequences x and y, denoted by $H(x, y)$, is the number of positions j such that $x_j \neq y_j$.*

Given a fully-labeled tree T, i.e., a tree in which each node v is labeled by a sequence s_v over some alphabet Σ, we define the Hamming distance of an edge $e \in E(T)$, denoted by $H(e)$, to be $H(s_u, s_v)$, where u and v are the two endpoints of e. We now define the parsimony score of a tree T.

Definition 2 *The parsimony score of a fully-labeled tree T, is $\sum_{e \in E(T)} H(e)$. Given a set S of sequences, a maximum parsimony tree for S is a tree leaf-labeled by S and assigned labels for the internal nodes, of minimum parsimony score.*

Given a set S of sequences, the MP problem is to find a maximum parsimony phylogenetic tree T for the set S. Unfortunately, this problem is NP-hard, even when the sequences are binary [1, 4]. One approach that is used in practice is to look at as many leaf-labeled trees as possible, and choose one with a minimum parsimony score. The problem of computing the parsimony score of a fixed leaf-labeled tree is solvable in polynomial time [3, 5].

In the context of phylogenetic networks, the evolutionary history of a single (non-recombining) gene is modeled by one of the trees contained inside the phylogenetic network of the species containing that gene. Therefore the evolutionary history of a site s is also modeled by a tree contained inside the phylogenetic network. A natural way to extend the tree-based parsimony score to fit a dataset that evolved on a network is to define the parsimony score for each site as the minimum parsimony score of that site over all trees contained inside the network.

Definition 3 *([6, 7, 9]) The parsimony score of a network N leaf-labeled by a set S of taxa, is*

$$NCost(N, S) := \sum_{s_i \in S} (\min_{T \in \mathcal{T}(N)} TCost(T, s_i))$$

where $TCost(T, s_i)$ is the parsimony score of site s_i on tree T.

Notice that as usually large segments of DNA, rather than single sites, evolve together, Definition 3 can be extended easily to reflect this fact, by partitioning the sequences S into non-overlapping blocks b_i of sites, rather than sites s_i, and replacing s_i by b_i in Definition 3. This extension may be very significant if, for example, the evolutionary history of a gene includes some recombination events, and hence that evolutionary history is not a single tree. In this case, the recombination breakpoint can be detected by experimenting with different block sizes.

Bibliography

[1] W.H.E. Day. Computationally difficult parsimony problems in phylogenetic systematics. *Journal of Theoretical Biology*, 103:429–438, 1983.

[2] J. Felsenstein. The newick tree format, 1986. http://evolution.genetics.washington.edu/phylip/newicktree.html.

[3] W. Fitch. Toward defining the course of evolution: minimum change for a specified tree topology. *Syst. Zool*, 20:406–416, 1971.

[4] L.R. Foulds and R.L. Graham. The steiner problem in phylogeny is NP-Complete. *Adv. Appl. Math.*, 3:43–49, 1982.

[5] J.A. Hartigan. Minimum mutation fits to a given tree. *Biometrics*, 29:53–65, 1973.

[6] J. Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Mathematical Biosciences*, 98:185–200, 1990.

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

[8] B.M.E. 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 Computational Biology and Bioinformatics*, 1(1):13–23, 2004.

[9] L. Nakhleh, G. Jin, F. Zhao, and J. Mellor-Crummey. Reconstructing phylogenetic networks using maximum parsimony. *Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference (CSB2005)*, pages 93–102, August 2005.

[10] L. Nakhleh, D. Ruths, and L.S. Wang. RIATA-HGT: A fast and accurate heuristic for reconstrucing horizontal gene transfer. In L. Wang, editor, *Proceedings of the Eleventh International Computing and Combinatorics Conference (COCOON 05)*, pages 84–93, 2005. LNCS #3595.

[11] L. Nakhleh, J. Sun, T. Warnow, R. Linder, B.M.E. Moret, and A. Tholse. Towards the development of computational tools for evaluating phylogenetic network reconstruction methods. In *Proc. 8th Pacific Symp. on Biocomputing (PSB03)*, pages 315–326. World Scientific Pub., 2003.

[12] L. Nakhleh and L.S. Wang. Phylogenetic networks: properties and relationship to trees and clusters. *LNCS Transactions on Computational Systems Biology II*, pages 82–99, 2005. LNBI #3680.

[13] L. Nakhleh and L.S. Wang. Phylogenetic networks, trees, and clusters. In *Proceedings of the 2005 International Workshop on Bioinformatics Research and Applications (IWBRA 05)*, pages 919–926, 2005. LNCS #3515.

[14] D.R. Robinson and L.R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131–147, 1981.

[15] D. Ruths and L. Nakhleh. Recombination and phylogeny: Effects and detection. *International Journal of Bioinformatics Research and Applications (IJBRA)*, 1(2):202–212, 2005.

[16] D. Ruths and L. Nakhleh. Recomp: A parsimony-based method for detecting recombination. In *Proc. 4th Asia-Pacific Bioinformatics Conference*, pages 59–68, 2006.

[17] M. Steel and T. Warnow. Kaikoura tree theorems: computing the maximum agreement subtree. *Information Processing Letters*, 48:77–82, 1993.

[18] C. Than, D. Ruths, and L. Nakhleh. Efficient enumeration of species/gene tree reconciliation scenarios. 2006. Under review.