NETGEN Release Notes

Kappa Version

(Phylogenetic Network Generation Application)

Monique M. Morin

Department of Computer Science
University of New Mexico

April 3, 2008

Abstract

NETGEN is an event-driven simulator that creates phylogenetic histories which include hybrids. The birth-death model typically employed by biologists is extended with a hybridization event (both diploid and polyploid hybrids are possible). DNA sequences are evolved in conjunction with the typology, enabling hybridization decisions to be based upon hamming distance if desired. NETGEN supports variable rate lineages, root sequence specification, outgroup generation and many other options. This document provides an overview of the software, installation and execution instructions, as well as a comprehensive list of input parameters.

Contents

1 Introduction

This document is intended to accompany the NETGEN software which is currently available for download at http://www.cs.unm.edu/∼morin/ and is released under GNU General Public License (GNU GPL).[1]

The software is a simulation tool for generating phylogenetic networks. The traditional birth-death model[2] used often in biology to create phylogenetic topologies is extended to have hybrids (either diploid, polyploid, or both) and/or variable rate lineages. Hybrid decisions can be made according to sequences associated with the lineages, which are developed in conjunction with the topology itself, or randomly.

This software is command line driven, written in C, and developed in a Debian Linux environment. Its operation on/in other platforms/environments has not been tried. Theoretically, interested parties should be able to use this software on any Linux/Unix platform which has a standard C/C++ compiler.

An application note describing a previous release of NETGEN has appeared in *Bioinformatics*; please cite the reference[3] and/or the download web page when using this software.

Caveat: The Network Generator software (NETGEN, comprised of the executables named NG and NSGW) is currently available as source from http://www.cs.unm.edu/∼morin/. The software is constantly under development as part of a research effort and is offered only "As-Is." There is no guarantee, written or implied, of the software being bug-free or reliable and no liability related to this software will be accepted.

[1] see http://www.gnu.org/licenses for license details

[2] see Rensahw, E., 1991, Modelling Biological Populations in Space and Time, Cambridge University Press.

[3] Morin and Moret, "NetGen: generating phylogenetic networks with diploid hybrids", Bioinformatics, Vol 22 Number 15, p1921-1923 August 2006.

2 Overview

NETGEN is developed with the intent of furthering phylogenetic network research. Our immediate goal is to develop a tool which can produce topologies and sequences for scenarios which include diploid and/or polyploid hybridization events occurring at an inter-species level.

We use a birth-death-hybrid model, where the birth-death portion is the traditional approach from biology which utilizes a Poisson model. We add "hybrid" as a third type of event (the type of hybridizations is then determined by "sub-rates"). The model is event driven and is continuous in nature which is appropriate for our inter-species perspective. As sequence knowledge is desired for making hybrid decisions, we employ SEQ-GEN[4] to simulate sequences for the lineages.

We also permit the birth, death, and hybrid rates to differ for each lineage[5], in order to permit the creation of a "variable rate model." As a separate parameter, the user can specify non-ultrametric networks where branch lengths are modified by applying a gamma distribution random variable. While constant-rate, ultrametric models are the standard in the field, we wanted to provide alternatives to both parameters as these assumptions are unrealistic under certain scenarios.

Due to these modeling choices, each lineage has its own set of rates (either applied identically in the constant version or based upon a gaussian in the variable rate version) and there exist two notions of branch length. The first is linked to "clock-time" which corresponds to the time which is followed by the simulation. The second is "evolutionary-time" which corresponds to the amount of evolutionary change which has occurred between events. Sequence evolution is more realistic with an *evolutionary* perspective of length, while a hybrid event requires the two species to be contemporary which is dictated by clock time.[6]

[4]Rambaut A. and Grassly N.C., **Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees**, Comput Appl Biosci 12, 235-238, 1997. See also http://evolve.zoo.ox.ac.uk for further information.

[5]the "sub-rates" to determine the type of hybridization remain constant

[6]In the ultrametric scenario, evolutionary branch length is calculated by scaling the clock-time length using a constant value derived from "expected evolutionary height" (an input value). In the

NETGEN maintains a queue of events which are processed according to their scheduled clock time. Currently the event types of: birth (speciation), death (extinction), and hybridization are implemented [7]. However, one can imagine adding other event types such as "lateral gene transfer" and/or global events which would modify wide-spread behavior (e.g. a mass extinction increasing the overall death rate). Event occurrences follow a Poisson model by having interarrival times drawn from an exponential distribution.[8]

Hamming distance is calculated as part of the hybridization decision process. In early versions of NETGEN, the hamming distance between two lineages was calculated as summing the difference found for each pairing of single strands of DNA across the lineages. For example, with two lineages (A and B), each having three strands, the hamming distance (hd) was calculated as hd(A_strand1, B_strand1) + hd(A_strand2, B_strand2) + hd(A_strand3, B_strand3), where hd is calculated in the normal manner of counting 1 for every base pair index that does not agree between the two strands/strings (e.g. hd(AGT, AGG) = 1).

However, with the addition of polyploid hybridizations, it was necessary to redefine hamming distance as it is possible that two lineages being examined may not agree in their number of homologous chromosomes (as a result of prior polyploidizations). Hence, hamming distance is now defined as an average value. The average is calculated by summing all pairwise comparisons of "equivalent" strands. Equivalent strands are defined by having the same strand and homologous chromosome number. The number of strands per homologous chromosome is fixed for the simulation so strand numbers are identical to the strand index in the code (as strand index is reset for each new chromosome). The homologous chromosome

non-ultrametric scenario, evolutionary branch length is additionally multiplied by a random value drawn from a gamma distribution to create evolutionary distance deviation.

[7] The type of hybridization (either diploid or polyploid) is determined when the event occurs

[8] A Poisson model provides a probability distribution based upon a rate (lambda). We know from statistics that events occurring based upon a Poisson model have interarrival times which follow an exponential distribution with the parameter lambda. For further details, see pages 159 and 163 of Hahn and Shapiro "Statistical Models in Engineering" and/or Ross "A First Course in Probability" page 165.

number however refers to the biological notion of a homologous chromosome where a lineage may have more than one of the same homologous chromosome (again as a result of polyploidization). For example, two lineages (one with 2 homologous chromosomes and one with 1 homologous chromosome – each with 1 strand) one would proceed as follows. Verifying that both lineages have the same number of "unique" homologous chromosomes (one in this case) the code calculates average hamming distance as (hd(A_chromo1_strand1, B_chromo1_strand1) + hd(A_chromo2_strand1, B_chromo1_strand1))/2. In the case of multiple strands per homologous chromosome, an average for each strand is found and then the average strand values are averaged for the homologous chromosome and all average distances for chromosomes are then averaged for an overall average hamming distance between the two lineages. Note that under this definition, the maximum average hamming distance is the length of a single sequence/strand (a value which is constant throughout the simulation).

An option for establishing an outgroup is provided in NETGEN. Sequences for an outgroup are assigned at the end of the simulation with a special routine which attempts to have the sequences *similar*, but not *too similar* to the other extant leaves of the network. Input parameters establish the range of similarity and also the level of effort that is given to generating these outgroup sequences.

Provided there are lineages active in the simulation, the simulation continues until a specified number of extant (current-day) taxa is reached. The clock time of when this is achieved is considered the "initial end time" and event processing stops. At this point, the simulation looks ahead to what would have been the next event and its associated time. An "official end time" then is randomly chosen between the "initial" and "next event" times. It is this "official end time" that the extant leaves (and outgroup if requested) will be assigned. This avoids the artificial result of having final branches with length zero and emulates biological reality in that current day taxon sampling occurs at a random point between two event times. If the simulation ends prematurely (due to no active lineages remaining),

the outgroup is not made and the desired number of extant taxa is not reached.[9]

3 Installation Guidelines

NETGEN is available as a compressed tar file. The directory structure is very specific and needs to maintained in order for the code to function properly. The following sections discuss the details of code location and dependencies.

3.1 Directory Structure

The tar file available for download has the required directory structure. The following depicts the basic hierarchy:

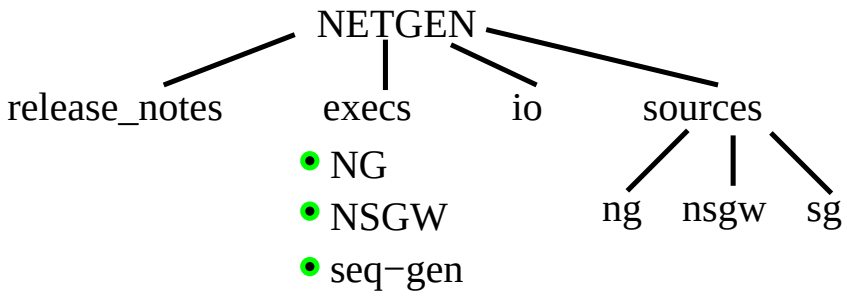

Figure 1: Required directory structure for NETGEN and related code. *NG*, *NSGW*, and *seq – gen* are names of distinct executables needed for this simulation tool.

3.2 Code Dependencies

NETGEN as a simulation tool is comprised of multiple executables – each of which has an independent use. However, for the purposes of this document, we assume the goal is to create phylogenetic networks with sequences, in which case the executables are used in a combined fashion.[10]

[9] A premature end to the simulation is reported in the output report.

[10] We will therefore not address how to use the executables separately at this time.

NETGEN is a simulator which creates a birth-death-hybrid topology. While the topology is evolving, sequences are needed for the lineages. Specifically, before a hybrid event takes place, all active lineages must have their sequences updated. In order to achieve this, NG makes a call to NSGW (Network Sequence Generator Wrapper) which handles the administrative work of calling and processing the results from SEQ-GEN (our chosen external sequence generator). [11]

Hence, the three executables `NG`, `NSGW`, and `seq-gen` are needed for a single run. The executables are kept in the *execs* directory (see Figure 1) and must be run from this same directory. There are intermediate files which are produced as the programs "talk" to each other and the intermediate files are placed here, each identified with the original process id (pid). Once a run is complete, these files can be deleted.[12] The files are named uniquely for each run in order to facilitate concurrent runs of NETGEN. It is recommended that the input and output files as specified as part of the command line are kept in the *io* directory, though this is not enforced or required as these locations are specified by the user at run time.

Our source code for `NG` and `NSGW` can be found in the *sources* directory. When these two sources are compiled using their respective Makefiles, the executables are automatically put into the *execs* directory. The running of NETGEN with sequences requires a copy of SEQ-GEN.[13] As currently implemented, the executable for SEQ-GEN must be placed in the *execs* directory and be named `seq-gen`. Currently SEQ-GEN VERSION 1.3.2 is being used. It is recommended that the copy of SEQ-GEN be kept in the *sources/sg/* subdirectory, but again this is not required or enforced. The file se_out_seq-gen_*pid* which can be found in the *execs* directory will contain output from the most recent call to SEQ-GEN and will contain the

[11] Currently, SEQ-GEN from the University of Oxford is used to generate sequences. However it is possible to use a different sequence generator by modifying the NSGW source code. (The NSGW code is designed as a customized piece of software acting as an interface between NG and a sequence generator.)

[12] The files are named se_out* for easy identification.

[13] The current version is available from the SEQ-GEN creators (see http://evolve.zoo.ox.ac.uk). A copy of SEQ-GEN version 1.3.2 (which we used for development) is posted on our web-site (http://www.phylo.unm.edu/∼morin/) for convenience.

version number of SEQ-GEN utilized. This file can also be deleted after the run is complete.

The random numbers used in all three executables are generated by code known as **Mersenne Twister**. This random number generator is known for its high periodicity and was thus chosen for NG and NSGW.[14,15]

4 Execution Guidelines

4.1 Execution Summary

The standard command line format for running NETGEN from the execs directory is:

- `./NG -i` *input_file* `-o` *output_file* `[-r` *unsigned_long*`]` `[>` *run_capture*`]`

For example:

- `./NG -i ../io/simple.in -o ../io/simple.out -r 84319 > ../io/simple.run_capture`

The first four items after the executable are required; the user must specify the input and output files in this format. The next parameter and its argument (-r and an unsigned long value) are optional allowing the user to specify a random number seed for repeatability purposes.[16] It is also recommended that STDOUT (standard out) is redirected as in the above format and example. All debugging and error messages are directed to STDOUT and it is best to capture these in a separate file.

[14]The creators provided .c and .h files for inclusion in code and more information can be found at http://www.math.sci.hiroshima-u.ac.jp/∼m-mat/MT/emt.html.

[15]M. Matsumoto and T. Nishimura, **Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator**, ACM Trans. on Modeling and Computer Simulation Vol. 8, No.1, January pp.3-30 (1998).

[16]While this is recommended it is not required and in the absence of a specified seed, the code will generate one based upon the processor clock.

4.2 Input File

The input file, which is specified on the command line, is a simple text file containing the parameters for a single simulation run. If a user desires multiple runs, a shell or perl script is recommended. (Most likely the user will want to establish and use a separate random number seed for each run. The piping of calls between the various executables are kept separate by attaching the process id to the intermediate files.) The following is a very simple input file and is included in the downloadable tar file as *simple.in*:

```
num_extant_taxa 6
outgroup
sp_rate 1.0
ex_rate 0.2
hyb_rate 0.5
simultaneous_sequences
init_sequence_length 10
modified_newick_report
splitstree_report
node_listing_report
```

4.3 Input Parameters

This section lists and describes parameters for the simulation which can be specified in an input file. Unless otherwise noted, each parameter (and any accompanying arguments) needs to be on a line by itself. Blank lines are skipped and any line starting with a # is considered a comment line and is ignored.

4.3.1 GENERAL PARAMETERS

- **num_extant_taxa** *x*

  where *x* is an integer which specifies the number of extant taxa; once this number of active lineages is reached by the simulation, the simulation will

halt; this value does not include an outgroup taxon if desired

- **outgroup**

  including this parameter will have the code generate an outgroup taxon (see Outgroup Related section below for further details about how the outgroup is created)

- **sp_rate** *x.x*

  where *x* is a real number which gives the instantaneous birth (speciation) rate

- **ex_rate** *x.x*

  where *x* is a real number which gives the instantaneous death (extinction) rate

- **hyb_rate** *x.x*

  where *x* is a real number which gives the instantaneous hybridization rate;

- **dip_rate** *x.x*

  where *x* is a real number which gives the percentage of hybrids which will be diploid in nature; the default if not specified is 1.0; this value when summed with the polyploid rate must equal 1.0

- **poly_rate** *x.x*

  where *x* is a real number which gives the percentage of hybrids which will be polyploid in nature; the default if not specified is 0.0; this value when summed with the diploid rate must equal 1.0

- **variable_rate**

  variable rate networks will be produced by applying varying speciation, extinction, and hybridization rates to every lineage; if this parameter is specified, each event rate (sp_rate, ex_rate, hyb_rate) will be taken as the mean of a normal distribution and variances for each can be specified as follows:

- **sp_rate_var** *x*
- **ex_rate_var** *y*
- **hyb_rate_var** *z*

where *x, y, z* are the variance values from the user; if variable rate is requested, but the individual variances are not provided by the user, each variance will have a default value of 1.0; (rate variations cannot be specified unless the variable_rate option is specified explicitly)

- **desired_height** *x*

  where *x* is a real number between 0 and 1; this value is used to calculate a scalar which is applied to every evolutionary branch length in the hope that the final height will be realistic with respect to evolutionary terms; this is provided as an option because it is believed biologists using a simulation such as NETGEN have a rough estimate as to how much evolutionary change is expected between the root and the extant taxa; the height of the network is not guaranteed to be the desired height as only an *expected* height can be calculated in advance of the simulation; note that this parameter does *not* affect the ultrametricity of a network as it is a constant scalar that is employed if chosen; if this parameter is not chosen, a default height scalar of 1 is used which has no effect on the evolutionary branch lengths

- **non_ultrametric** *x y*

  if this parameter is *not* specified, the default is that the network generated will be ultrametric (all extant taxa will be equi-distant from the root with respect to evolutionary branch lengths); if specified, a non-ultrametric network will be produced by multiplying each evolutionary branch length by a gamma random variable; *x* is the shape value for gamma and *must* be an integer, while *y* is the scale variable for gamma

4.3.2 SEQUENCE RELATED

NOTE: If neither of the following two parameters is explicitly stated in the input file, the default is simultaneous_sequences.

- **simultaneous_sequences**

  if this is specified, NSGW will be called by NG to generate sequences during the topology creation

- **no_simultaneous_sequences**

  this option is intended to be used for quick execution of NETGEN when only trees (no hybridization events) are desired; sequences will not be generated under this option; if hybrids are desired, and this option is specified,[17] a method other than hamming distance, must be used to choose the second parent (see the Hybrid Related section for further details)

- **init_sequence_length** *m*

  where *m* is an integer value specifying how long each of the root sequences should be

- **seq_gen_options** "*string*"

  where "*string*" (including the quotes) is used as input to NSGW which passes it along as input to SEQ-GEN; this allows the user to specify things such as which model is used by SEQ-GEN; the argument is *not* sanity checked – it must be a legitimate string to give SEQ-GEN; note that in newer versions (including 1.3.2) of SEQ-GEN there cannot be a space between *-m* and the model name, so *"-m HKY"*, will fail and *"-mHKY"* is needed

- **num_root_strands_per_homologous_chromosome** *n*

  where *n* is an integer value to specify how many strands the root should have (default is 2 if not specified); however if sequences are given (see below) the

[17] this combination is implemented for algorithmic research purposes and is not intended for common use

user must provide num_root_strands_per_hc with a number so the input routine will know how many sequences to read; note that num_root_strands_per_homogous_chromosomes * num_root_homologous_chromosomes is the number of sequences that will be associated with the root

- **num_root_homologous_chromosomes** *n*

  where *n* is an integer value to specify how many chromosomes the root should have (default is 1 if not specified); however if sequences are given (see below) the user must provide num_root_homologous_chromosomes with a number so the input routine will know how many sequences to read; note also it is assumed the sequences are grouped by homologous strands for each chromosome

- **initial_root_sequences**

  AGCT... (chromosome 0 / strand 0)

  AGCT... (chromosome 0 / strand 1)

  AGCT... (chromosome 0 / strand 2)

  ...

  AGCT... (chromosome x / strand 0)

  AGCT... (chromosome x / strand 1)

  ...

  lets the user specify exactly what the starting sequences are (if not specified, a sequence will be generated randomly and copied into each of the root's sequences as the assumption is that initially the root sequences are identical); can only be used if num_root_homologous_chromosomes and num_root_strands_per_homologous_chromos have been previously declared

4.3.3 HYBRID RELATED

- **max_num_hybrids** *m*

where *m* is an integer limiting the number of hybrids created during the simulation

- **hyb_ev_dist_threshold** *x.x*

  for use with the hyb_parent_min_ev_distance option for selecting second parent discussed below; this is a real value of a threshold for which two lineages can hybridize (evolutionary distance must be less than or equal to this value); the default (if the parameter is not specified, is DBL_MAX which results in effectively no threshold being applied)

- **hyb_exp_fn_threshold_rate** *0.x*

  for use with the hyb_parent_exp_fn option for selecting second parent discussed below; this is a real value between 0 and 1 which is multiplied by the max hamming distance (the sequence length) – the resulting value determines the boundary for which two lineages can hybridize; the default for the parameter if not specified is .20;

- **hyb_ham_threshold_rate** *0.x*

  for use with the hyb_parent_min_hamming option for selecting second parent discussed below; this is a real value which multiplied with max hamming distance (sequence length) (and truncated if needed to make an integer value) dictates the threshold for which two lineages with sequences can hybridize (average hamming distance must be less than or equal to this value); the default, if this parameter is not specified, is 1.0 (resulting in effectively no threshold being applied)

- **hyb_parent_min_hamming**

  this parameter specifies that the second parent of a hybrid will be chosen by finding a second parent with minimal average hamming distance from the first parent (which also meets the hamming threshold as discussed above); if there are multiple potential second parents one is chosen randomly; this is the default for choosing the second parent if another method is not specified

- **hyb_parent_min_ev_distance**

  this parameter specifies that the second parent of a hybrid will be chosen by finding a second parent with minimal evolutionary distance from the first parent (which also meets the distance threshold as discussed above); if there are multiple potential second parents with the same distance, one is chosen randomly;

- **hyb_parent_exp_fn** *x*

  this parameter specifies that the second parent of a hybrid will be chosen according to an exponential function; *x* is the average hamming distance which has $1/e$ probability of being the second parent's distance from the first parent (declaring *x* defines an exponential function); *x* must be at least 1 and less than or equal to the max hamming distance possible (sequence length); when it is time to choose a second parent, a random value *y* from the defined exponential distribution is chosen; all active lineages having average *y* hamming distance from the first parent are identified and one of those candidates is chosen randomly; if there are no active lineages having *y* distance, the search is expanded incrementally by increasing/decreasing the value of *y* by 1 (until the `hyb_exp_fn_threshold` discussed above is reached) and searching for candidates and repeating the process;)

The following three parameters are not intended for common use. They are implemented for algorithmic research purposes and are included here only for the sake of completeness. When one of these parameters is included, sequences and the threshold do not play a role in the simulation.

- **hyb_parent_random**

  this parameter specifies that the second parent of a hybrid will be chosen randomly from all active lineages; this permits exploration of any biasing which occurs with hamming distance based selection

- **hyb_parent_min_bl**

this parameter specifies that the second parent of a hybrid will be chosen by finding the active node with the smallest branch length; this permits further exploration of any biasing of how second parents are chosen

- **hyb_parent_max_bl**

  this parameter specifies that the second parent of a hybrid will be chosen by finding the active node with the longest branch length; this permits further exploration of any biasing of how second parents are chosen

4.3.4 OUTGROUP RELATED

- **min_outgroup_diff_value** *m*

  allows user to specify an integer value of the minimum hamming distance between the outgroup and each leaf – the hamming distance here is calculated at the strand/sequence level

- **max_outgroup_diff_value** *n*

  allows user to specify an integer value of the maximum hamming distance between the outgroup and each leaf – the hamming distance here is calculated at the strand/sequence level

- **min_outgroup_diff_perc** *0.x*

  to be used in place of *min_outgroup_diff_perc*; allows user to specify a percentage of the initial sequence length to describe the minimum hamming distance between the outgroup and each leaf – hamming distance here is calculated at the strand/sequence level

- **max_outgroup_diff_perc** *0.y*

  to be used in place of *max_outgroup_diff_perc*; allows user to specify a percentage of the initial sequence length to describe the maximum hamming distance between the outgroup and each leaf – hamming distance here is calculated at the strand/sequence level

- **max_outgroup_tries** *m*

  integer value to cap the maximum number of tries when looking for an outgroup which meets the above bounds

4.3.5 OPTIONAL REPORTS

All optional reports have each line started with a `c` (for comment). This allows for easy identification and parsing of the output file as a whole. See the Output Report section for some details about these reports.

- **modified_newick_report**

  if specified, either the Newick Format (for a tree) or our own *Modified* Newick format (for a network–see next section) will be printed at the end of the output file

- **node_listing_report**

  if specified a report sequentially listing all nodes (and their sequences if simulated) will be printed at the end of the output file

- **hhd_from_root_report**

  if specified a report listing each diploid hybrid's hamming distance from the root will be printed at the end of the output file

- **splitstree_report**

  if specified a report compatible with the SPLITSTREE software (see the next section) will be printed at the end of the output file

4.3.6 MISCELLANEOUS

The following are for debugging and validation purposes only. They are included here for completeness, but it is not expected that they will ever be used by the average user.

- **report_branch_hd**

  if specified, this will produce a report that gives an edge listing for all edges that includes the hamming distance from the node to its parent as well as other branch information; this report is used to examine the hamming distance for each edge, which is not expected to be the same as the evolutionary distance as sites can reverse themselves along a single branch; this report can be used for the typical run and/or in conjunction with the next option

- **update_seqs_for_all_events**

  this option forces the intermediate update of branch lengths and sequences for all events (birth, death, and hybridization) wherease normally this only occurs for hybrid events; when this option is invoked, the previous option should also be used so a report at the branch length level is provided – note that the intermediate steps are not reported, but a comparison of branch length hamming distances for when run with and without this option is meaningful in showing that sequences from SEQ-GEN change linearly; note that this significantly slows down the simulation process and should be used rarely with small sizes

- **hamming_edge_report**

  if specified a VERY lengthy report concerning edge lengths and hamming distances (at the sequence level) and how they relate during more than one "generation" will be printed at the end of the output file

- **track_nsgw_seeds**

  if specified, the seeds given to NSGW (up to 10,000 of them) will be saved (as well as the corresponding node id for which NSGW is being called); this information is printed as part of the final report for testing purposes

- **fake_branch_lengths**

  if this is specified, fake_branch_lengths_yn in the code is set to true and the FakeBranchLength routine will be used to assign hard coded branch lengths

to all branches

4.4 Output Reports

The primary output report is given the name specified by the user on the command line (see Section 4.1). Basic run parameters are reported along with a listing of branches, nodes, and their affiliated sequences. The file contains comment lines, starting with the letter C, which explain the output.

The parsimony score of the network is reported on a comment line. The parsimony score is the sum of all the evolutionary changes (as calculated by hamming distance) along each branch of the network for every single strand sequence. Every strand can be followed down the network thus the calculation of this measure is straightforward. (Hybrid nodes are either comprised of partial sequences from each parent (as in the case of diploid hybridizations) or all sequences from each parent (as in the case of polyploid hybridization).) Figures 2 and 3 illustrate the idea for both types of hybrids.

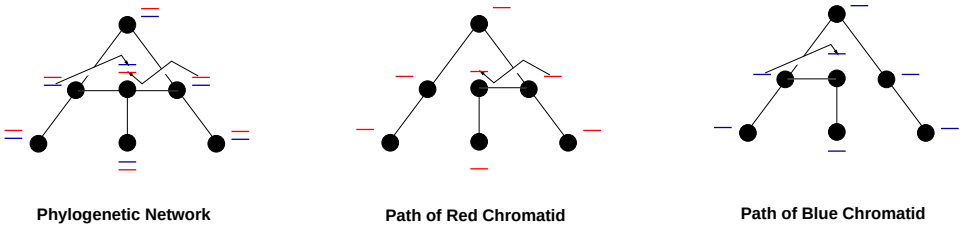

Figure 2: Parsimony score idea (diploid hybrid) – trace each root strand down the network and measure the evolutionary change along the way.

Additional reports can be requested (see the previous section). Each line of these additional reports also starts with the letter C so to avoid input confusion when using this file/report as independent input into a program such as NSGW. Typically if these additional reports are to be used, the user will want to copy the appropriate section into a separate file and remove the leading C character before proceeding.

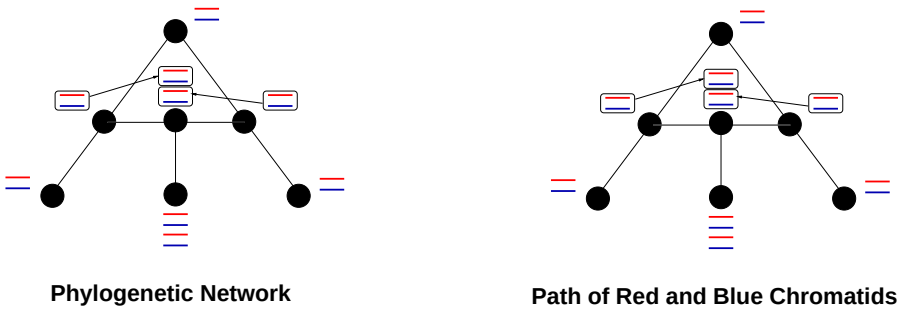

Figure 3: Parsimony score idea (polyploid hybrid) – trace each root strand down the network and measure the evolutionary change along the way – note that that the path of evolution for the strands is the same as the network in this case because all sequences are used for a polyploid hybrid.

The `modified_newick_report` is a custom version of the well accepted Newick format. If NETGEN has produced a tree, this report will generate the standard Newick format. If however a network has been created, the modified version, identifying hybrid nodes with `#H` will be outputted. Likewise `#E` is used to annotate extant leaves. This format includes all information needed to recreate the topology including evolutionary branch lengths. It is hoped that software which accepts and displays the Newick format for trees will be adapted for this extension. Figure 4 shows a simple example of the original Newick format and modified version.

The `splitstree_report` generates a network format for the software SPLITSTREE.[18] After the removal of the first line (`c SplitsTree Format`) and the leading `c` character of each line, the report can be read by the latest version of SPLITSTREE. This is a possibility for viewing the network topology. One must realize this does not include branch lengths (either clock or evolutionary) and the default depiction is not a rooted network, though SPLITSTREE allows the user to

[18]Daniel H. Huson and David Bryant, **Application of Phylogenetic Networks in Evolutionary Studies**, Molecular Biology and Evolution 23(2):254-267, 2006; http://www-ab.informatik.uni-tuebingen.de/software/jsplits/welcome.html

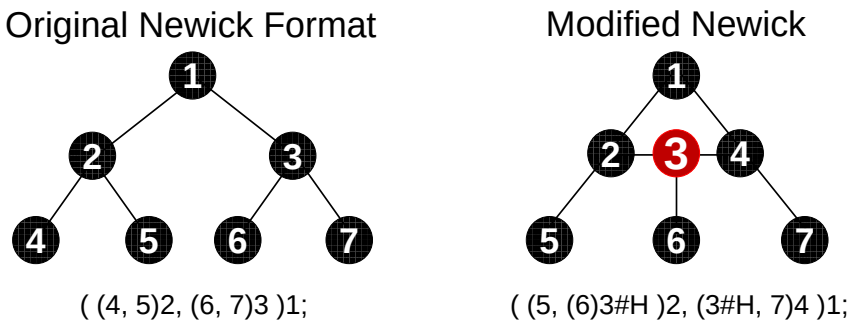

Figure 4: Original versus Modified Newick formats. Phylogeny on the right contains the hybrid node (3) denoted by *#H* in the text string.

manually rearrange vertices.

5 Acknowledgements and Contact

Special thanks to my advisor Bernard Moret for multiple in-depth conversations about the model and simulator. This work has been supported by the National Science Foundation under grants IIS 01-21377, DEB 01-20709, and EF 03-31654.

The author of this software and document can be contacted by the following means:

morin@cs.unm.edu

OR

Monique Morin
University of New Mexico
Department of Computer Science
Mail stop: MSC01 1130
1 University of New Mexico
Albuquerque, NM 87131-0001
USA