Design Document
Gene Tree Improvement Tool: Tree Fix

May15-10

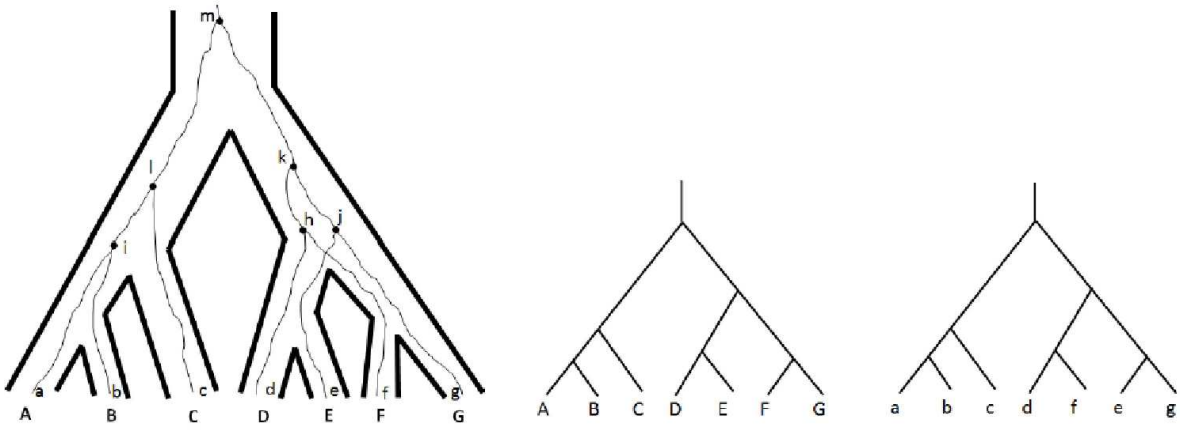
Ian Ray
Paul Leichty
Cole Poffenberger
Ben Streit

1 Project Statement

TreeFix is a command line application from a computational biology group at MIT. It allows users to evaluate gene trees, by determining the duplication loss cost (duploss). The duplication and loss model determines the number of duplication and loss events required to fit the given gene tree in the given species tree. Given a gene tree, the related species tree, and an alignment of DNA sequences associated with genetree, the program then attempts to offer an improved gene tree topology. This feasibility is grounded on the basis that 1) the existential statistical likelihood of the resultant gene tree is equal to, or greater than, that of the input gene tree, and 2) the duploss cost is the minimal choice between the resultant tree and input tree.

This software tool is written in Python, and is supported with a group of C functions. We intend to take the existing form of TreeFix and extend it to evaluate gene trees using the expanded Robinson-Foulds “MulRF” cost, as well as to take non-binary gene and species trees as inputs. The MulRF cost method differs from that of the duploss cost method mainly because while the duploss is a biological cost the Robinson-Foulds distance is a mathematical cost. Both are useful and the user should have a choice of which cost method to use.

Figure 1: pictures of a species tree, gene tree and their combination



(a) Gene tree in the species tree (b) The species tree (c) The gene tree

2 Goals

MulRF is a cost evaluation mechanism, in the same category as the duploss cost model. It differs, however, in that it wants to take “mul-trees,” or in this case, non-binary trees (trees where any nodes can possess 3 or more children). MulRF is an expansion on the Robinson-Foulds (RF) distance of a gene tree to its species tree. By default, RF and duploss

cost evaluation metrics expect binary tree inputs for gene and species trees. However, only the MulRf cost may be computed for *non-binary trees* as well.

The purpose of this expansion lies in the domain of application for each cost evaluation mechanism. Simply put, *duploss cost,* TreeFix's default cost metric, is one solution of many to the problem; the RF cost will provide a more accurate analysis in some cases. Our project's aim will *amplify TreeFix to make it more robust.* Our formal goal definitions to achieve improved robustness are:

1. **Extend Treefix to include MulRF algorithm for calculating tree distance.**
   Interestingly, if you hand a MulRF cost evaluation metric all binary trees, it effectively computes the RF. By extending treefix to allow for the use of MulRF, we therefore also extending it to allow RF evaluations.
2. **Allow Treefix to take in non-binary species trees as inputs.**
   This, of course, is necessary to calculate RF on mul-trees, also known as the MulRF cost evaluation.

3 Deliverables

1. A TreeFix package containing additional and restructured code, which adds a command line option to use MulRF. If the MulRF option is selected, tool produces a result based on the Robinson Foulds/MulRF distance.
2. A Github repo containing documentation, bugs tracking, and the finished software tool.

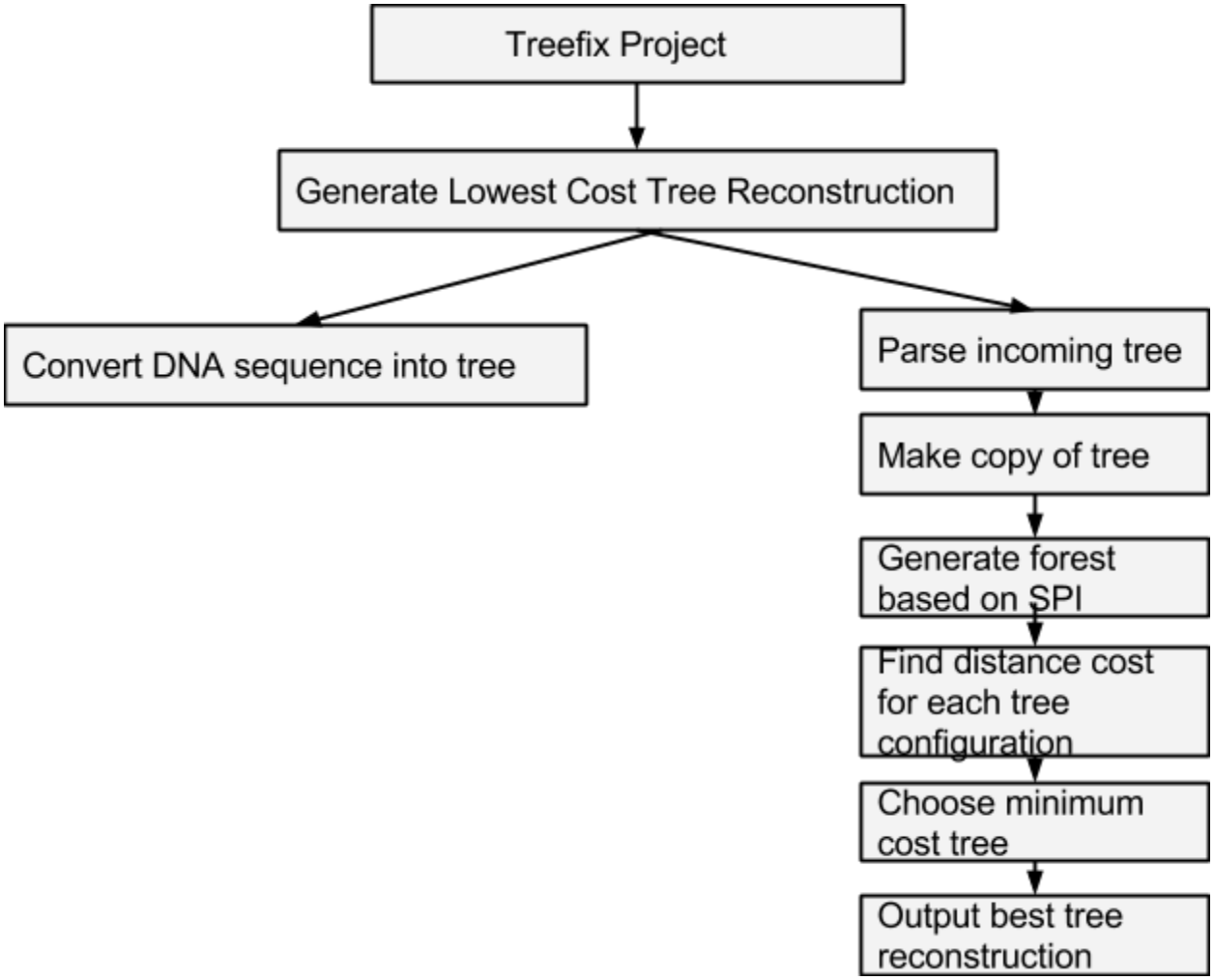
---

4 System Level Design

4.1 System Requirements

1. Python (2.5.4 or greater): http://python.org
2. C compiler (gcc)
3. SWIG (1.3.29 or greater): http://www.swig.org
4. Numpy (1.5.1 or greater): http://www.numpy.org
5. Scipy (0.7.1 or greater): http://www.scipy.org

4.2 Functional Decomposition



4.3 System Analysis

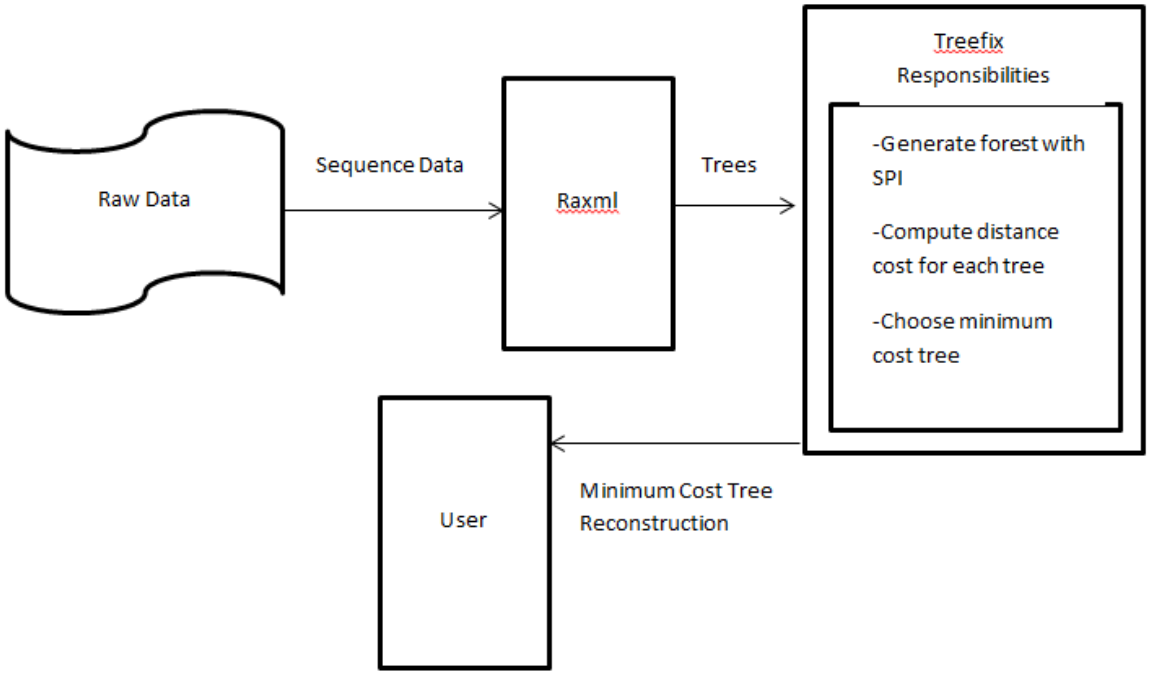
4.3.1 Feasibility

Implementing different cost modules is a very straight forward. If there are any difficulties it will be in how to adapt the existing formats for the already implemented modules. This becomes a little more difficult for our group due to the code needing to be modified being written in python, a language none of us have very much, if any,

personal experience in. This will slow down the groups progress but should not hinder the results of the project.

Where the project may become infeasible is with the clients desire to be able to have non-binary trees be output by TreeFix. This is currently handled by the raxml package, which we believe already has the ability to input non-binary trees, yet the outputs will always be binary. Many test cases will need to be ran to make sure this is the case along with some research into the raxml package files. If it turns out we have to completely change how the raxml package behaves we do not know if that will be feasible given both our groups time and lack of knowledge on how the trees correlate to each other.

4.4 Block Diagram



4.5 Detailed Description

The following outline addresses program file hierarchies and extensions. It contains classes, denoted by black square-bullets, that will be inserted into existing modules as extensions.

These classes must be added in order to enable the correct use of the MulRF cost model. Black circle-bullets represent folders at the root level of the treefix program folder, empty circle-bullets represent python modules (also technically folders) within those folders. MulRFCostModel is an extension of an existing class CostModel that requires the implementation of four functions; they are listed. We will also need to reimplement the program's main function to allow a MulRF command line flag option. It is not listed here, while it is trivial work.

- treefix
  - Models
    - MulRFModel(CostModel)
      - _init_(self, extra)
        - required method(like a constructor in java), will be used to initialize the cost model and creates parser options for the class
      - optimize_model(self, gtree, stree, gene2species)
        - guarantee that the gene tree and species tree are in their most optimal formats to reconcile, this includes making sure that the tree is rooted.
      - recon_root(self, gtree, newCopy=True, returnCost=False)
        - reroot the tree by minimizing the robinson foulds distance
      - compute_cost(stree, gtree)
        - will compute the Robinson Foulds distance by calling a method within the dep.compbio.phylo class
  - dep
    - compbio
      - phylo
        - Existing code will be black boxed but the following methods will need to be added to the class
        - Label_events_MulRF()
          - May not be necessary, but if needed will label the nodes with their cluster
        - countCostMulRF()
          - will compute the Robinson Foulds distance by comparing clusters and finding the number of clusters without a pair
    - rasmus
      - treelib
        - Existing code will be black boxed, MulRF Class will need to use treelib but should no requirements to the existing code

4.6 I/O Specifications

Input: Treefix includes several command line arguments to specify the input. Here is the list of these arguments:

-s <species tree>, --stree=<species tree>
specifies the location of the species tree file (in newick format)

-S <species map>, --smap=<species map>
specifies the location of the file mapping gene names to species names

-A <alignment file extension>, --alignext=<alignment file extension>
alignment file extension (default: ".align")

-o <old tree file extension>, --oldext=<old tree file extension>
old tree file extension (default: ".tree")

-n <new tree file extension>, --newext=<new tree file extension>
file extension for the file where the reconstructed gene tree will be written

-l <log file>, --log=<log file>
log filename. Use '-' to display on stdout.

-V <verbosity level>, --verbose=<verbosity level>
verbosity level of the log file (0=quiet, 1=low, 2=medium, 3=high

-e <extra arguments to module>, --extra=<extra arguments to module>
extra arguments to pass to the program that computes likelihoods

--niter=<# iterations>
number of search iterations to be performed

--smodule <cost model>
specifies which cost module to use (duploss or MulRF)

Output: After generating the improved gene trees, Treefix will output it's gene trees in newick format as a .tree file.

4.7 Interface Specifications

Our client has specifically requested that no GUI be implemented, while the command-line console is entirely sufficient at this time. The gene/species trees must be presented to the tool in Newick tree format. More information on Newick format here: http://en.wikipedia.org/wiki/Newick_format.

Action	Interface
Input species tree	console
Output Best Cost Gene Tree	no interface, outputs tree in a .tree file to the working directory.

4.8 Hardware Specifications

There are no hardware specifications for this project, while it is entirely software-domain.

4.9 Software Specifications

4.9.1 Functional Requirements

- To accept both binary and nonbinary trees
- Can calculate multiple types of cost algorithms.

4.9.2 Nonfunctional Requirements

- Can easily be extended to include new reconciliation cost algorithms (modularized, well documented)
- Easily accessible and compilable
    - Easy, rich access will come in the form of a github repository
    - instructions on compilation/installation will be included
- No modification to the interface (as requested by client)

5 Testing

5.1 Simulations And Modeling

Each simulation will take a preconstructed tree input into the system given to us by our client Dr Burleigh. These will allow for us to understand the accuracy of our results and to ensure our implementations are running correctly.

5.2 Implementations Issues/Challenges

1. Implementing an accurate and efficient MulRF cost model.
2. Integrating MulRF cost model into existing code.

3. Refactoring TreeFix to accept non-binary trees.

5.3 Testing, Procedures and Specifications

Testing will proceed in two phases: In the first phase, the programmers will make simple unit tests on the program, to move the program towards preparation for the final testing phase.

Procedure One:
During initial testing/debugging phases, procedure one will be used and defined as:

1. **Ensure that program compiles, runs without error, produces "promising result".** A promising result is any result that is non-trivial or clearly non-false. For example, if an empty graph is produced, we have produced a false result.
2. **Tester will commence testing in test domain.** Tester may be any one of the group members. The testing domain will be treefix with debugging verbosity enabled, as well as a relevant input for the test. This relevant input may be a trivial input, or a complex input whose result we are certain of.
3. **Take Feedback on results.** The quality of results, as determined by the tester, will direct the next course of action before the next testing cycle.

Procedure Two:
When final product testing is feasible, necessary, or useful, procedure two will be used/defined as:

1. **Ensure that program compiles, runs without error, produces "promising result".** A promising result is any result that is non-trivial or clearly non-false. For example, if an empty graph is produced, we have produced a false result.
2. **Zip program, send to advisor and client.** For the foreseeable future, the measured correctness of data created by tool can only come from experts in the Field, i.e. Ruchi Chaudhary (our advisor) and Gordon Burleigh (our client).
3. **Take Feedback on results.** The quality of results, as determined by our data experts (advisor, client), will direct the next course of action before the next testing cycle.

In regard to step 3 of both cases:
If we deem it useful, a private bug repository will be managed for the duration of the project.

6 Conclusion

The completion of this project certainly seems feasible within two semesters. In terms of hefty targets that must be hit in order to successfully deliver our final product, there is but one: the correct implementation of the MulRF cost model class. Of course, hitting this target has its own set of dependencies, and it is also a fairly *broad* target. Furthermore, the

underestimation of a task's difficulty carries with it inherently problematic ramifications. Given our certainty that we can hit our main target, our concern shifts to answering the question of "how do we guarantee a successful implementation of MulRF's class?"

We have already found that the act of placing high value on *communication, organization,* and *perseverance* can deliver outstanding results. At the onset of the project, the vastness of the program's files and libraries seemed daunting. This told us we should develop a good heuristic for pruning out modules that would not change. Ultimately, focusing on these three principles enabled us to identify specific modules that would need to be reimplemented, and pre-existing library functions that are reusable in that implementation. If we consider these values in all of our future project decisions, we believe with certainty that MulRF-TreeFix will arrive on or before schedule.

7 Definitions

Duploss: Duplication and loss is a biological cost; this cost is the number of evolutionary events, i.e., duplications and losses in genes, required to fit the gene tree in the species tree.

Gene tree: an evolutionary tree of individual genes.

MulRF: MulRF distance extends the RF distance so it can be computed for the tree pair in which one tree is multi-copy but the other is not. Gene trees are multi-copy most of the time. (Multi-copy tree means a tree in which multiple leaves can have the same label.) So the regular RF distance can not always compute the distance between a gene tree and a species tree pair, but MulRF can.

RF: or Robinson-Foulds distance is a mathematical cost, computed by counting number of bipartitions present in one tree but not in the other. RF distance is defined for the tree pair on the identical leaf label set (NOT multi-set).

Species tree: an acyclic graph (tree) representing evolutionary relation between species