Introduction

This program is a tool for assessing conflict in homologous gene trees without losing any data due to duplication events in those gene trees. This can be achieved by decomposing each homologous gene tree into a set of subtrees. Each subtree contains no repeating taxa, but the full set of subtrees constitutes a complete and non-overlapping representation of every speciation node in the homologous gene tree. This allows every speciation node in every gene tree to be assessed for conflict and concordance with the species tree using a bipartition-based method.

The program has four modes: normal (n), reverse (r), search (s), and summarize (summarize). Normal maps conflict from a folder of gene trees back onto a species tree; reverse maps conflict with a given species tree onto a gene tree; search allows searching for a specific bipartition of interest in a collection of gene trees and summarize takes an annotated gene tree from reverse mode and describes some of the general features of that tree. All of the modes (except summarize) use the same method to generate subtrees from homologous gene trees, outlined below.

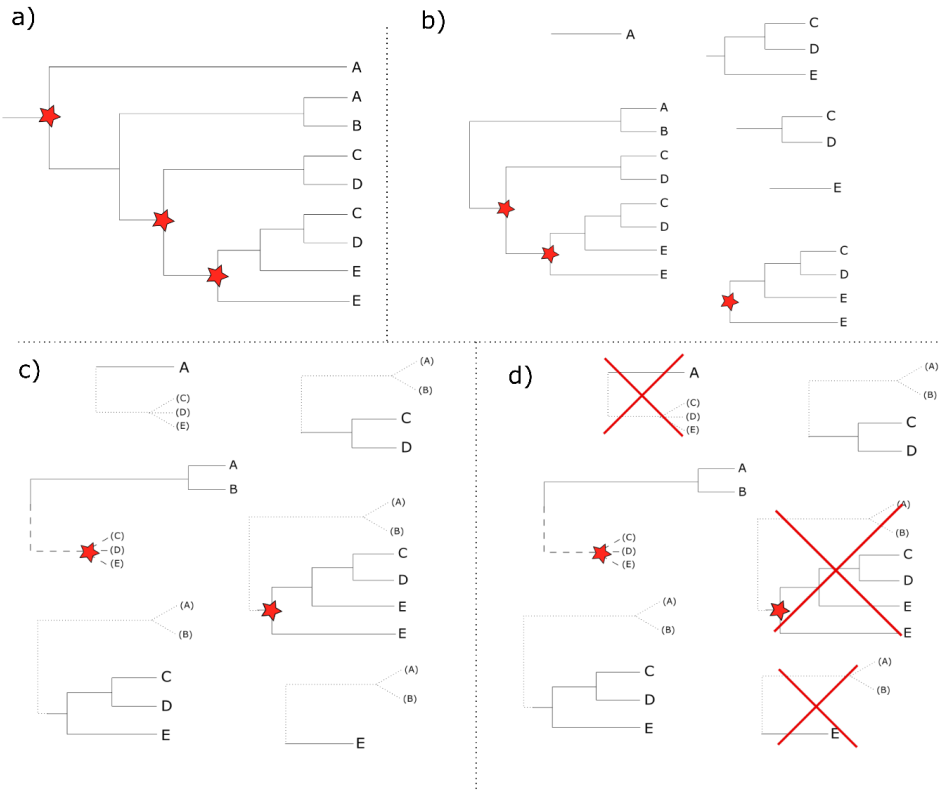

Figure 1: The steps required for subtree decomposition and subsequent processing. First, the homolog tree, containing homologs from 5 distinct species (A, B, C, D, and E), is decomposed into preliminary subtrees by splitting at each duplication event (red stars), with every child node of every duplication in the tree being added to the subtree list (a, b). Then, any duplications located within each of these subtrees (except at the root node) are collapsed and marked as not to be analysed (dashed branch as in the leftmost centre subtree in (c)). The outgroups of each subtree are also included but labelled as not to be analysed (dotted branches) to enable searching for missing taxa (c). Finally, any subtrees whose root node is a duplication or leads directly to a tip are excluded, leaving only the informative subtrees with no overlap.

N.B.: The default prefix for all files generated by this program is *out*, and this is the naming convention used here, but *out* can be replaced with any other prefix if desired using the optional argument *--outfile_prefix*.
N.B. 2: A complete list of all optional arguments for this program is given at the end of this manual.

Normal mode ('n' mode)
After the subtrees have been generated in normal mode, they are used to create bipartitions, then compared with the species tree bipartitions and used to assess conflict and concordance. Each identified conflict or concordance is mapped back onto the corresponding node on the species tree. The final sums of conflicts and concordances are reported in the outfiles *out_conflict.tre* and *out_concord.tre,* respectively. Normal mode also generates several other outfiles, including a more detailed list of all the conflicts (*out_analysis.csv*). A complete list of all these files is shown at the end of this section.

Example input
For a simple analysis, using the prefix test1 for all output files (outfile prefix defaults to "out"):
```
python2 scripts/conflict_detector.py --mode n --species_tree
path/to/species_tree.tre --gene_folder path/to/gene_folder/ --
outfile_prefix test1
```

For a default analysis using a cutoff (nodes with a label below the cutoff will not be analysed; nodes with no label at all will still be included):
```
python2 scripts/conflict_detector.py --mode n --species_tree
path/to/species_tree.tre --gene_folder path/to/gene_folder/ --cutoff
80
```

For an analysis where detailed information on the conflicts and concordances is not required:
```
python2 scripts/conflict_detector.py --mode n --species_tree
path/to/species_tree.tre --gene_folder path/to/gene_folder/ --no_csv
```

For a very detailed analysis, generating a file that contains every subtree generated in the analysis:
```
python2 scripts/conflict_detector.py --mode n --species_tree
path/to/species_tree.tre --gene_folder path/to/gene_folder/ --
output_subtrees
```

Example output
If *--outfile_prefix* is specified, "out" in each of these filenames will be replaced by the specified prefix (e.g., *out_analysis.csv* would become *test1_analysis.csv* if the first example input above was used).

out_analysis.csv
(optional, can be removed using the --no_csv argument)

This file contains all of the conflicts found with the species tree. The first column (node_id) lists the node in the species tree at which the conflict was found. The species_bipart and

ortholog_biparts columns list the relevant bipartition in the species tree and the conflict bipartition in the gene tree, respectively. Occasionally, two nodes in a subtree both conflict with the same node in the species tree without being nested in each other. If this was the case, one of the two conflicts is reported in the ortholog_bipart column and the other in the alternative_conflict column. Otherwise, this column is empty. The number_of_conflicts column shows the total number of times each conflict occurred (including if it occurred multiple times in the same homolog tree). The percentage column indicates that number as a percentage of the total number of conflicts at that node in the species tree. Finally, the genes column lists the name of every file where that specific conflict occurred.

out_concord_genes.csv
This file contains a list of all the nodes on the species tree (column one) and all of the gene trees that have concordances with that node (column two). Each gene tree is only listed once for each node, even if that tree contained more than one concordance with said node.

out_conflict_genes.csv
As above but for conflict.

out_multi_concord_gene_counts.csv
This file contains records of all the times a gene tree was concordant with the species tree and which node it was concordant with. Each row corresponds to a specific gene tree and node. For example, row 1 might be node 4, gene tree 1; row 2 might be node 4, gene tree 2, and row 3 might be node 8, gene tree 1. Column one contains the species tree node label, and column two contains the gene tree filename, copied as many times as that tree was concordant with the species tree.

out_multi_conflict_gene_counts.csv
As above but for conflict.

out_concord.tre
A copy of the species tree where each node is labelled with the total number of recorded concordances across all gene trees.

out_conflict.tre
A copy of the species tree where each node is labelled with the total number of recorded conflicts across all gene trees.

out_concord_gene_counts.tre
A copy of the species tree where each node is labelled with the number of gene trees that were concordant with that node.

out_conflict_gene_counts.tre
A copy of the species tree where each node is labelled with the number of gene trees that conflicted with that node.

out_labels.tre
A copy of the species tree where each node has a unique numerical label (node_id). It is used to cross-reference with the output tables (.csv files) listed above.

out_total_analyzed.tre
A copy of the species tree listing the total number of times each node was analyzed and found to be in **either** conflict **or** concordance. This is only output when the cutoff is not set because when a cutoff is set, nodes below that cutoff are never analysed and do not count towards the total number of relationships.

Reverse mode ('r' mode)
In reverse mode, conflicts and concordances generated in normal mode are mapped back onto a single gene tree instead of the species tree. Reverse mode only generates one outfile, called *out_concon.tre* (or equivalent prefix in place of "out"). This is a gene tree in newick format where each node is labelled as either in conflict (X), concordant (*), a duplication (D), or uninformative (U) with respect to the species tree.

Example input
For a simple analysis:
```
python2 scripts/conflict_detector.py --mode r --species_tree
path/to/species_tree.tre --gene_tree path/to/gene_tree.tre
```

The arguments *--cutoff*, *--no_csv,* and *--output_subtrees* can be used in the same way as the normal mode for reverse mode.

Search mode ('s' mode)
In search mode, a folder of gene trees can be searched for a user-specified query bipartition, and the program will return a file called *out_search.csv* (or equivalent prefix in place of "out"). This is a file containing: in column one, the filename of each gene tree where a match with the query was found, and, in column two, the number of times within that gene tree the query matched. The program also prints an output displaying the total number of gene families and the total number of matches.

Example input
For a simple analysis:
```
python2 scripts/conflict_detector.py --mode s --gene_folder
path/to/gene/folder/ --query_bipart taxon1,taxon2,taxon3
```

For an analysis where some taxa may be excluded from the query bipartition by the program if they are missing from the tree being searched:
```
python2 scripts/conflict_detector.py --mode s --gene_folder
path/to/gene/folder/ --query_bipart taxon1,taxon2,taxon3 --
query_remove taxon2
```

The arguments *--cutoff* and *--output_subtrees* can be used in the same way as the normal mode for search mode.

Summarize mode ('summarize' mode)
Summarize mode analyses the annotated gene tree produced by reverse mode.

When only supplied with an annotated gene tree, summarize mode prints five statistics about the tree to stdout: the total numbers of duplications, concordances, conflicts, uninformative nodes, and tips (taxa).

When supplied with an annotated gene tree and a species tree and query bipart, summarize mode prints the same five statistics and prints other statistics to an output file called *out.tsv* (or a different specified prefix). This tab-separated table lists the features found downstream of the query in each gene tree.

Example input
For a simple analysis:
```
python2 scripts/conflict_detector.py --mode summarize --
annotated_tree out_concon.tre
```

For a more detailed analysis that can be found in *out.tsv*:
```
python2 scripts/conflict_detector.py --mode summarize --
annotated_tree out_concon.tre --query_bipart taxon1,taxon2 --
species_tree path/to/species_tree.tre
```

Full argument list for the program

-d, --description
Gives a short description of the program and its different modes.

--species_tree
A file containing the species tree. Required in 'n' and 'r' modes. All trees should be in newick format.

--mode
The mode that we run the program in. It should be n (normal), r (reverse), s (search) or summarize (for analysing the output of reverse).

--gene_tree
File containing a gene tree. Required in 'r' mode. All trees should be in newick format.

--gene_folder
A folder containing all the gene trees to be analysed. Required for 'n' and 's' modes. All trees should be in newick format.

--cutoff
Nodes with a bootstrap label below the cutoff value will not be included in the analysis.

--query_bipart
A query to be searched. Required in 's' mode, should be entered in the form: 'taxon1,taxon2', with no spaces.

--query_remove

Comma separated taxa that can be excluded from the query bipartition (only 's' mode). This does not account for tree structure!

--no_csv
The program does not create a .csv file with a detailed breakdown of the different conflict abundances in normal mode.

--outfile_prefix
A prefix for all of the outfiles produced by the program in whichever mode you run it in.

--output_subtrees
Will output subtrees analyzed to a file that has the gene name followed by .subtree.

--annotated_tree
File containing a gene tree labelled by 'r' mode. Required in 'summarize' mode.

Summarizing the results with Pie.py

Pie.py is provided as a method for generating pie charts from the results. This will generate a series of .svg files containing piecharts that correspond to the nodes reported for the _labels.tre. This can be done for both when you have run the program on orthologs (same as phyparts) or can be done when the program has been run on homolog trees.

Options

--help
Brings up the help menu.

--folder
The location of all the input files - can be a full path or just from current working directory.

--input_prefix
The prefix of input files (X, where X is X_labels.tre, X_concord.tre, etc).

--totals_tree
The *_total_analyzed.tre file that corresponds. To have this you will need to have run the analysis without a support cutoff. The total analyzed has every time a node was analyzed, so if you want the pie chart to show how many were uninformative based upon support values you will need to include the total analyzed file.

--alternatives_reported
The number of alternative conflicts to be reported, anything up to 15. If you specify three it will report the concordant relationship,

along with the most common conflict, the second most common conflict and the third most common conflict as separate colors.

--output_type
Should the pie chart be labelled (l), emphasized (e), both (le) or neither (default)?

--alt_total
This is for if you used orthologs for the analysis. The program will then infer that all nodes were possible to be analyzed based upon the number of orthologs that were available.

Example of how to run Pies.py

This is a command for if you ran the program but ignored support values. The piecharts produced will be both emphasized and labelled with which conflict the color is. The color key for the conflicts will also be printed to the terminal screen.

```
python3 Pie.py -f Test/ -p CarnivoryNoSup -t
Test/CarnivoryNoSup_total_analyzed.tre -o le -a 3
```

This is a command for if you ran the program but ignored support values. The piecharts produced will be both emphasized and labelled with which conflict the color is. The color key for the conflicts will also be printed to the terminal screen.

To perform the same procedure but factoring uninformative you can run:

```
python3 Pie.py -f Test_Sup/ -p CarnivoryWithSup -t
Test/CarnivoryNoSup_total_analyzed.tre -o le -a 3
```