

RIATAHGT Command

Description

Detects and reconstructs horizontal gene transfer events from phylogenetic incongruence. The input trees must be specified in the [Rich Newick Format](#).

In brief, the algorithm reads in a single species tree and any number of gene trees. It returns the horizontal gene transfer events. The tool performs some algorithmic techniques to speed up the detection of HGT. One technique is to partition the solutions into independent subsets, and the tool then finds equivalent solutions to for every subset. In this way, the tool decreases the size, and thus decreases the running time, of the search space. The second technique is to handle (non-binary) caterpillars by replacing them with a chain of three leaves. In general, species and gene trees can be non-binary. There are various reasons for this, such as tree reconstruction errors, insufficient information to build trees. The tool maximally refines trees so that it does not introduce new HGT before detecting HGT.

The tool prints solutions for each pair of species tree and gene tree. 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 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 output more concise and informative. From the tool's output, one can get a complete solution by selecting a subsolution 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:70.0,b:75.0):90.0,c:80.0):60.0,(((e:80.0,f:75.0):95.0, g:60.0):80.0,d:70.0));
```

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

```
-----  
Component I5:  
Subsolution1:  
I2 -> d (70.0)  
Subsolution2:  
d -> I2 (80.0)  
Subsolution3:  
I5 -> I4 (70.0) [time violation?]  
-----  
Component I2:  
Subsolution1:  
f -> e (95.0)  
Subsolution2:  
I2 -> g (95.0) [time violation?]  
Subsolution3:  
e -> f (95.0)  
-----  
Component I4:  
Subsolution1:  
b -> a (90.0)  
I4  
Subsolution2:  
I4 -> c (90.0) [time violation?]  
Subsolution3:  
a -> b (90.0)  
*****
```

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 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 the edge incident into `sn` to the edge incident into `tn` in the species tree.

If the branches in the gene tree has bootstrap values, the tool infers a support value for HGT events. The support value for each HGT event is in parentheses next it. For example, event `f -> e` in component `I2` has support 95%. In the case the species tree branches also have bootstrap values, they are also taken into account when the tool computes the support value for HGT.

The tool also computes a support value for HGT it detected, based on the bootstrap values of branches in the gene tree (and the species tree, if those values are present). A key advantage of the method is that it does not require resampling gene sequences, which greatly reduces running time.

The above output format is compact, and so it is very useful for presentation. However, the tool also allows you to display the full solutions, by using the option `-e`. In this case, the tool will present solutions in the form of a network in eNewick format.

Usage

```
RIATAHGT species_tree_ident (gene_tree_ident1 [, gene_tree_ident2...]) [-u] [-p prefix] [-e] [result output file]
```

| | | |
|---|---|-----------|
| {species_tree_ident} | The input species tree identifier. | mandatory |
| {gene_tree_ident1 [, gene_tree_ident2 ...]} | Comma delimited list of gene tree identifiers. See details . | mandatory |
| -u | Prevent the trees from being refined and contracted. | optional |
| -p prefix | Specifies a name prefix used to label internal nodes. If no prefix is specified, then the default prefix is used. | optional |
| -e | Present full solutions. | optional |
| result output file | Optional file destination for command output. | optional |

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.

Examples

```
#NEXUS

BEGIN TREES;

Tree speceiesTree = ((e,(f,g):0.63:.70)::.80,((a,(b,c):0.5:.80):0.5:.67,d):0.6);
Tree geneTree1 = (((a,b):0.5:.70,c):0.6:.80,(d,((e,f):0.4:.70,g)::.70)::.80);
Tree geneTree2 = ((e,(f,g):0.5:.70):0.6:.80,((a,b):0.5:.90,(c,d):0.5:.87):0.57:.72);

END;

BEGIN PHYLONET;

RIATAHGT speceiesTree (geneTree1, geneTree2);

END
```

Command References

- L. Nakhleh, D. Ruths, and L.S. Wang. RIATA-HGT: A fast and accurate heuristic for reconstructing horizontal gene transfer. In L. Wang, editor, *Proceedings of the Eleventh International Computing and Combinatorics Conference (COCOON 05)*, pages 84–93, 2005. LNCS #3595.
- C. Than, G. Jin, and L. Nakhleh. Integrating sequence and topology for efficient and accurate detection of horizontal gene transfer. In *Proceedings of the Sixth RECOMB Comparative Genomics Satellite Workshop. Lecture Notes in Bioinformatics (LNBI #5267)*, pages 113–127, 2008.
- C. Than and L. Nakhleh. SPR-based tree reconciliation: Non-binary trees and multiple solutions. In *Proceedings of the Sixth Asia Pacific Bioinformatics Conference (APBC)*, 2008.

See Also

- [List of PhyloNet Commands](#)