Using INC within Divide-and-Conquer Phylogeny Estimation (Supplementary Materials)

Thien Le[1], Aaron Sy[2], Erin K. Molloy[1], Qiuyi (Richard) Zhang[3], Satish Rao[2], and Tandy Warnow[1]

[1]Department of Computer Science , University of Illinois at Urbana-Champaign , 201 S. Goodwin Ave, Urbana, IL 61801 , USA

[2]Department of EECS , University of California at Berkeley, Berkeley CA 94720, USA

[3] Department of Mathematics , University of California at Berkeley, Berkeley CA 94720, USA,

1 Implementation details

We try different distance functions to generate the distance matrix d; but the default setting uses pairwise log-det distance [7]

Constraint trees are computed based on some recursive decomposition of an initial tree on all taxa of $\mathcal{A}$. Details are as followed

1. Start with an initial tree t_0 and a number $m > 0$.
2. Initialize the forest $F = \{t_0\}$.
3. Repeat until F is no longer changed: for all $t \in F$, if the leaf set of t has size at least m, decompose t into t_1 and t_2 using centroid edge decomposition, update $F := F \backslash \{t\} \cup \{t_1, t_2\}$.

This is dubbed the 'PASTA decomposition' for its use in the multiple sequence alignment software PASTA [3].

The end result of this procedure is a forest of leaf-disjoint trees $F = \{t_1, \ldots, t_k\}$. We explore 2 options:

1. Use F directly as input to constrained INC.
2. For each $i \in 1 \ldots k$, let S_i be the sub-alignment on the leaves of $t_i \in F$, then for each S_i, run a maximum likelihood tree inference method (either FastTree2 or RAxML) on it to generate tree t'_i. Use $F' = \{t'_i, i \in 1 \ldots k\}$ as input to constrained INC.

The default setting uses the second option and employs FastTree2 as the ML heuristic.

1.1 Minimum Spanning Tree (MST) calculation

Within constrained INC, the first step is to compute an MST from input distance matrix d. The purpose of this step is twofold: to get the maximum edge weight of the MST and to get the tree-building ordering of the taxa. The theory of constrained INC mandates that at any time of the

tree-building process, the graph induced by the leaf set of the growing tree is connected in the MST.

For this purpose, we employ a heap implementation of Prim's algorithm. This algorithm maintains a connected tree as it builds the MST so we can use the order in which taxa are added to the MST as the tree growing order for tree-building in constrained INC. We maintain the longest edge added, the ordering, as well as the immediate parent of each taxon as we compute the MST.

1.2 Valid subtree identification

As one adds some query taxon q to the growing tree, constrained INC restricts the subtree upon which q may be added, dubbed the valid subtree for q. This subtree depends on the constraint tree that q comes from; and by adding q only to edges within this subtree, the final output is guaranteed to agrees with all constraint tree.

Let t_g, t_c be the growing tree and the constraint tree containing q. For some tree t and some set A of taxa, let t^A be the induced tree defined similarly to [9]. For a set A of taxa, an unrooted tree t and a taxon $p \notin A$, let $LCA(A, t, q)$ be lowest common ancestors of A in the tree t when t is rooted at q. The exact implementation for the valid subtree identification is as follow

1. Find A the intersection of t_g's and t_c's leaf sets.
2. Partition A into A_1, A_2 such that removing the immediate parent of q in $t_c^{A\cup\{q\}}$ makes the tripartition $\{\{q\}, A_1, A_2\}$. This can be done by 3 DFS procedures. The first DFS finds $l = LCA(A, t_c, q)$. Let l_1 and l_2 be the 2 children of l (t_c is still rooted at q). Then the second and third DFS find all leaves of the subtree rooted at l_1 and l_2 and mark them as belonging to A_1 or A_2 respectively.
3. Map l_1 and l_2 to t_g. This can be done by 2 DFS procedures. Let a_1 and a_2 be some taxon in A_1 and A_2 respectively (which exists due to the way we define A_1, A_2). The first DFS finds $l_1' = LCA(A_1, t_g, a_2)$ and the second finds $l_2' = LCA(A_2, t_g, a_1)$.
4. Find which neighbor of l_1' leads to l_2'. This can be done by 1 DFS procedure that stops when l_2' is reached and backtrack to get the correct neighbor.

2 Checking that output trees satisfies constraint trees

After each run, a script is run to make sure that output tree satisfies all constraint trees (as guaranteed by the algorithm). The script identify all constraint trees; extract the set of taxa in the constraint tree and build the subtree containing exactly those taxa in the output tree. Finally, it tests whether this subtree is identical to the constraint tree (RF distance 0).

The exact commands are

```
nw_prune -v <output_tree> $(cat <constraint_tree_label>) > <subtree_name>
compare_tree.py <subtree_name> <constraint_tree>
```

3 Specific commands

Some commands that are used throughout the study:

3.1 FastTree-2 [5] under GTRGAMMA model

```
FastTree -nt -gtr -gamma -quiet < <input_alignment> > <output_tree>
```

3.2 FastTree-2 [5] under JC

```
FastTree -nt -quiet < <input_alignment> > <output_tree>
```

3.3 FastTree-2 [5] initial tree

```
FastTree -nt -quiet -noml -nni 0 -log <log_file> < <input_alignment>
```

3.4 RAxML [6] under GTRGAMMA

```
raxmlHPC-PTHREADS-SSE3 -T 12 --silent -m GTRCAT -V --JC69
    -j -n <output_tree> -s <input_alignment>
    -w <working_directory> -p <random_seed> > <output_log>
```

3.5 RAxML [6] under JC

```
raxmlHPC-PTHREADS-SSE3 -T 12 --silent -m GTRGAMMA
    -j -n <output_tree> -s <input_alignment>
    -w <working_directory> -p <random_seed> > <output_log>
```

3.6 PAUP* [8] NJ using distance matrix

```
echo "ToNEXUS format=PHYLIP fromFile=<distance_matrix
    toFile=<temporaries_nexus_file>
    dataType=distance dmatrix=full;
    exe <temporaries_nexus_file>; NJ distance=K2P showtree=No;
    savetrees file=<output_tree> format=newick;" | ./paup4a164_centos64 -n
```

3.7 PAUP* [8] NJ using alignment under log-det distance

```
echo "ToNEXUS format=FASTA fromFile=<fasta_alignment>
    toFile=<temporaries_nexus_file>;
    exe <temporaries_nexus_file>; NJ distance=logDet showtree=No;
    savetrees file=<output_tree> format=newick;" | ./paup4a164_centos64 -n
```

3.8 FastME [2]

```
fastme -i <input_distance_matrix> -o <output_tree> -m B -n -s
```

3.9 nw_prune [1] (compute an induced binary subtree containing a given set of leaves)

```
nw_prune -v <output_tree> $(cat <constraint_tree_label>) > <subtree_name>
```

3.10 compare_tree.py [4] (computing RF distance between 2 trees)

```
python compare_tree.py <estimated_tree>  <reference_tree>
```

3.11 tree_to_dist.py (compute pairwise matrix distance under additive metric of the given tree)

```
tree_to_dist.py -t <input_tree> > <output_distance_matrix>
```

3.12 build_subsets_from_tree.py [4] (PASTA decomposition)

```
build_subsets_from_tree.py -t <initial_tree> -o <output_prefix>
    -n <max_subset_size>
```

3.13 remove_internal_labels.py [4](tree sanitization)

```
remove_internal_labels.py -t <input_tree> -o <output_tree>
```

3.14 PAUP* [8] distance matrix calculation

```
echo  "ToNEXUS format=FASTA fromFile=<input_alignment>
    toFile=<temporaries_nexus_file>;
    exe <temporaries_nexus_file>; DSet distance=K2P;
    SaveDist format=PHYLIP file=<output_distance_matrix>
    triangle=both diagonal=yes;" | ./paup4a14_centos64 -n
```

References

[1] T. Junier and E. M. Zdobnov. The newick utilities: high-throughput phylogenetic tree processing in the unix shell. Bioinformatics, 26(13):1669–1670, 2010.

[2] V. Lefort, R. Desper, and O. Gascuel. FastME 2.0: A comprehensive, accurate, and fast distance-based phylogeny inference program. Molecular Biology and Evolution, 32(10):2798–2800, 2015.

[3] S. Mirarab, N. Nguyen, L.-S. Wang, S. Guo, J. Kim, and T. Warnow. PASTA: ultra-large multiple sequence alignment of nucleotide and amino acid sequences. J. Computational Biology, 22:377–386, 2015.

[4] E. K. Molloy and T. Warnow. NJMerge: A Generic Technique for Scaling Phylogeny Estimation Methods and Its Application to Species Trees. In M. Blanchette and A. Ouangraoua, editors, Comparative Genomics. RECOMB-CG 2018. Lecture Notes in Computer Science, volume 11183. Springer, Cham, 2018.

[5] M. N. Price, P. S. Dehal, and A. P. Arkin. FastTree 2 - Approximately Maximum-Likelihood Trees for Large Alignments. PLOS ONE, 5(3):1–10, 2010.

[6] A. Stamatakis. RAxML version 8: A tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics, 2014.

[7] M. A. Steel. Recovering a tree from the leaf colourations it generates under a Markov model. Applied Mathematics Letters, 7(2):19 – 23, 1994.

[8] D. L. Swofford. PAUP* (*Phylogenetic Analysis Using PAUP), Version 4a161, 2018.

[9] Q. Zhang, S. Rao, and T. Warnow. Constrained incremental tree building: new absolute fast converging phylogeny estimation methods with improved scalability and accuracy. Algorithms for Molecular Biology, 14(2), 2019.