Bioinformatics, 34(19), 2018, 3399–3401
doi: 10.1093/bioinformatics/bty384
Advance Access Publication Date: 8 May 2018
Applications Note

Phylogenetics

Two C++ libraries for counting trees on a phylogenetic terrace

R. Biczok[1,†], P. Bozsoky[1,†], P. Eisenmann[1,†], J. Ernst[1,†], T. Ribizel[1,†], F. Scholz[1,†], A. Trefzer[1,†], F. Weber[1,†], M. Hamann[1] and A. Stamatakis[1,2,*]

[1]Institute for Theoretical Informatics, Karlsruhe Institute of Technology, Karlsruhe 76128, Germany and [2]Scientific Computing Group, Heidelberg Institute for Theoretical Studies, Heidelberg 69118, Germany

*To whom correspondence should be addressed.
†The authors wish it to be known that, in their opinion, the first eight authors should be regarded as Joint First Authors.
Associate Editor: Russell Schwartz

Received on November 6, 2017; revised on May 1, 2018; editorial decision on May 2, 2018; accepted on May 3, 2018

Abstract

Motivation: The presence of terraces in phylogenetic tree space, i.e. a potentially large number of distinct tree topologies that have *exactly* the same analytical likelihood score, was first described by Sanderson *et al.* However, popular software tools for maximum likelihood and Bayesian phylogenetic inference do not yet routinely report, if inferred phylogenies reside on a terrace, or not. We believe, this is due to the lack of an efficient library to (i) determine if a tree resides on a terrace, (ii) calculate how many trees reside on a terrace and (iii) enumerate all trees on a terrace.
Results: In our bioinformatics practical that is set up as a programming contest we developed two efficient and independent C++ implementations of the SUPERB algorithm by Constantinescu and Sankoff (1995) for counting and enumerating trees on a terrace. Both implementations yield *exactly* the same results, are more than one order of magnitude faster, and require one order of magnitude less memory than a previous thirrd party python implementation.
Availability and implementation: The source codes are available under GNU GPL at https://github.com/terraphast.
Contact: alexandros.stamatakis@h-its.org
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

It is common practice to infer phylogenies on multi-gene datasets. One way to analyze these is to concatenate the data from several genes or entire genomes into one large super-matrix and infer a phylogeny on it via maximum likelihood (ML) or Bayesian inference methods. Typically, the sites of such a super-matrix are grouped into p disjoint partitions (e.g. genes) $P_1, ..., P_p$. Each partition is assumed to evolve according to an independent model of evolution and has a separate set of likelihood model parameters (e.g. substitution rates, branch lengths etc.).

Super-matrices often exhibit patches of missing data as sequence data for a specific taxon might not be available for *all* partitions P_i. Such patches occur because a specific taxon might simply not contain a gene/partition or because the gene has not been sequenced yet. In partitioned datasets, patches of missing data can induce an important effect on the likelihood scores of trees. Under specific partitioning schemes, model settings, and patterns of missing data, topologically distinct trees might have exactly the same analytical likelihood score if they reside on a terrace. Two distinct trees reside on a terrace if the sets of their induced partition trees are identical. Recognizing terraces, determining their size, and enumerating all trees on a terrace therefore constitutes an important step when searching tree space but also for post-processing the results of phylogenetic analyses. Final output trees of tree searches can reside on a terrace and thus, represent only *one* of many possible solutions.

The presence of terraces in likelihood-based inferences was first used implicitly by Stamatakis and Alachiotis (2010) to accelerate ML calculations. One year later, the terrace phenomenon was explicitly named and mathematically characterized by Sanderson et al. (2011). Additional properties of terraces, in particular their impact on bootstrap and other support measures were discussed by Sanderson et al. (2015). Chernomor et al. (2015, 2016) presented production-level implementations of topological moves that detect if consecutive trees reside on a terrace and thereby save computations in ML tree searches. Finally, D. Zwickl developed a python tool called terraphy for detecting terraces (https://github.com/zwickl/terraphy) based on the algorithm by Constantinescu and Sankoff (1995).

2 Implementation

2.1 Interface

The C and C++ interfaces (see https://github.com/terraphast) take as input: a Newick tree string; a binary matrix M of size $n \times p$, where n is the number of taxa and p the number of partitions and where every row is annotated by a corresponding taxon name, that denotes if data are available or not for species i at partition j; a bitmask specifying the computation mode (tree on a terrace; number of trees on terrace; enumeration of all trees on terrace); a destination file pointer to potentially print out all trees on the terrace; a pointer to a big integer library object for storing the number of terraces. For the latter we use the GNU multiple precision arithmetic library (GMP) by default to prevent integer overflow. The interface function returns an integer that either contains an error code or indicates a successful invocation.

2.2 Limitations

As the library calculates the number of unrooted trees on a terrace given an unrooted, strictly bifurcating input tree, the following limitation applies: the binary input matrix must contain at least one row without any missing data, a so-called comprehensive taxon tax_C such that all p induced per-partition trees $T|P_i$ can be consistently rooted on the branch leading to tax_C (for an approach to relax the requirement for a comprehensive taxon, see Supplementary Material). By induced per-partition tree, we refer to the input tree pruned down to the taxa for which sequence data *are* available at a partition i. This limitation allows to execute the SUPERB algorithm and, as we show in the supplement, guarantees that the number of rooted trees on the terrace calculated by SUPERB is identical to the number of unrooted trees on the terrace. This limitation can be circumvented by including an appropriate comprehensive outgroup sequence from a reference genome into the dataset.

3 Results

We initially tested our implementations on several artificial small five-taxon datasets for which either all possible trees reside on a single terrace or no terrace exists.

Subsequently, we tested both implementations on 26 empirical datasets from recently published biological studies (available at https://github.com/BDobrin/data.sets) and compared their performance to terraphy. For empirical datasets that did not contain a comprehensive taxon, we sub-sampled partitions such that the samples *did* contain a comprehensive taxon. For our tests we used a reference system with four physical Intel i7-2600 cores running at 3.40 GHz and with 16-GB main memory. We first verified that our two completely independent implementations (Terraphast I and II) yield exactly the same results and also compared their run-time performance to terraphy. Under identical

Table 1. Sequential execution times (seconds) for counting trees on a terrace with terraphy and Terraphast I/II

Dataset	Terraphy	Terraphast I	Terraphast II	Terrace size
Rosaceae	2.32	0.033	0.087	1.72×10^{23}
Shi.bats	6.34	0.015	0.081	2.42×10^{35}
Burleigh.small	4099.76	147.74	301.09	4.12×10^{50}

settings (see Supplementary Material for details), all three codes yielded exactly the same number of unrooted trees on *all* datasets, provided that the input tree is rooted at the same comprehensive taxon tax_C.

In Table 1 we provide the average sequential execution times over 10 runs and number of trees on the respective terrace for Terraphast I and II and terraphy on the three empirical datasets with the largest terraces. All three codes were executed in tree counting mode, that is, enumeration and printout of all topologies on the terrace was disabled. Additional computational experiments under different modes, including memory utilization, parallel performance, and additional empirical datasets as well as a discussion of the reasons for the performance difference between Terraphasts I and II are provided in the Supplementary Material. We recommend use of terraphast I as it is faster and also actively developed as well as maintained in a separate repository (https://github.com/upsj/terraphast-one).

4 Conclusions

We have provided two independent C++ implementations of the SUPERB algorithm for counting trees on a phylogenetic terrace. Because we developed two independent implementations that yield exactly identical results, we are confident that the implementations are correct. Furthermore, Terraphast I is 28 times faster than terraphy on the dataset containing the largest terrace (Burleigh.small) and requires one order of magnitude less RAM (see Supplementary Table S2). As our experiments with empirical datasets show, a plethora of published phylogenetic trees *do* reside on a terrace. Although the phenomenon has been known since 2011, authors of empirical studies do not routinely assess if their tree resides on a terrace. We are optimistic that the availability of an efficient and easy-to-integrate library for this purpose will facilitate integration of this important phylogenetic post-processing step into popular phylogenetic inference tools that are predominantly written in C or C++. terraphast I has already been integrated into RAxML-NG (https://github.com/amkozlov/raxml-ng). The authors of GARLI (D. Zwickl, personal communication, October 2017) and IQ-Tree (B.Q. Minh, personal communication, October 2017) also intend to integrate it into their tools.

Acknowledgements

We thank O. Chernomor, B.Q. Minh, D. Zwickl for discussions on the interface, B. Dobrin for access to her empirical dataset repository, A. Kozlov for integration with RAxML-NG, and the support by the state of Baden-Württemberg via bwHPC.

Funding

This work was financially supported by the Klaus Tschira Foundation and DFG [grant WA 654/22-2].

Conflict of Interest: none declared.

References

Chernomor,O. *et al*. (2015) Consequences of common topological rearrangements for partition trees in phylogenomic inference. *J. Comput. Biol*., **22**, 1129–1142.

Chernomor,O. *et al*. (2016) Terrace aware data structure for phylogenomic inference from supermatrices. *Syst. Biol*., **65**, 997–1008.

Constantinescu,M., and Sankoff,D. (1995) An efficient algorithm for supertrees. *J. Classification*, **12**, 101–112.

Sanderson,M.J. *et al*. (2011) Terraces in phylogenetic tree space. *Science*, **333**, 448–450.

Sanderson,M.J. *et al*. (2015) Impacts of terraces on phylogenetic inference. *Syst. Biol*., **64**, 709–726.

Stamatakis,A., and Alachiotis,N. (2010) Time and memory efficient likelihood-based tree searches on phylogenomic alignments with missing data. *Bioinformatics*, **26**, i132–i139.