Bioinformatics, 36(7), 2020, 2280–2281
doi: 10.1093/bioinformatics/btz874
Advance Access Publication Date: 22 November 2019
Applications Note

Phylogenetics

A fast and memory-efficient implementation of the transfer bootstrap

Sarah Lutteropp[1,*], Alexey M. Kozlov[1] and Alexandros Stamatakis[1,2]

[1]Computational Molecular Evolution Group, Heidelberg Institute for Theoretical Studies, Heidelberg 69118 and [2]Institute for Theoretical Informatics, Karlsruhe Institute of Technology, Karlsruhe 76128, Germany

*To whom correspondence should be addressed.

Associate Editor: Russell Schwartz

Received on August 23, 2019; revised on November 7, 2019; editorial decision on November 12, 2019; accepted on November 21, 2019

Abstract

Motivation: Recently, Lemoine *et al.* suggested the transfer bootstrap expectation (TBE) branch support metric as an alternative to classical phylogenetic bootstrap support for taxon-rich datasets. However, the original TBE implementation in the `booster` tool is compute- and memory-intensive.

Results: We developed a fast and memory-efficient TBE implementation. We improve upon the original algorithm by Lemoine *et al.* via several algorithmic and technical optimizations. On empirical as well as on random tree sets with varying taxon counts, our implementation is up to 480 times faster than `booster`. Furthermore, it only requires memory that is linear in the number of taxa, which leads to 10× to 40× memory savings compared with `booster`.

Availability and implementation: Our implementation has been partially integrated into `pll-modules` and `RAxML-NG` and is available under the GNU Affero General Public License v3.0 at https://github.com/ddarriba/pll-modules and https://github.com/amkozlov/raxml-ng. The parallel version that also computes additional TBE-related statistics is available at: https://github.com/lutteropp/raxml-ng/tree/tbe.

Contact: sarah.lutteropp@h-its.org

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The Felsenstein bootstrap (BS) (FBP) procedure (Felsenstein, 1985) is widely used to assess the robustness of phylogenies. The FBP draws columns from the multiple sequence alignment (MSA) with replacement to generate 100 or more MSA replicates. Then, for each MSA replicate, a corresponding BS tree is inferred. Subsequently, the BS support value of a branch in the reference tree (e.g. the best-known ML tree on the original MSA) is computed by calculating the frequency of occurrence of this branch/bipartition in the BS trees. In the classical FBP approach, only bipartitions that match *exactly* are counted. Conversely, the transfer BS expectation (TBE) metric (Lemoine *et al.*, 2018) also takes into account all 'similar' bipartitions in the BS replicate trees. The contribution of such similar bipartitions is weighted by their similarity to the respective reference bipartition.

TBE support computations are based on the so-called transfer distance. The transfer distance $\delta(b, b^*)$ between a branch b in the reference tree and a branch b^* in a BS replicate is the minimum number of taxa that need to be moved to transform the bipartition induced by b into the bipartition induced by b^*. The transfer index $\phi(b, T^*)$ is defined as the minimum transfer distance between a branch b in the reference tree and the branches in the BS replicate tree T^*:

$$\phi(b, T^*) = \min_{b^* \in T^*} \delta(b, b^*).$$

Given a reference tree and a set of BS replicate trees, Lemoine *et al.* define the TBE(b) of a branch b in the reference tree as follows:

$$\mathrm{TBE}(b) = 1 - \frac{\overline{\phi(b, T^*)}}{p - 1},$$

where $\overline{\phi(b, T^*)}$ is the average transfer index over all BS replicates and p is the number of taxa on the 'light' side of the bipartition induced by b. The part of a bipartition that contains the smaller tip set is referred to as the 'light side' of the bipartition, whereas the larger tip set is called the 'heavy side'. When both sets are of equal size, the 'light side' is chosen arbitrarily.

2 Implementation

We implemented the transfer BS computation as part of the `pll-modules` library. The `pll-modules` library offers high-level functions (e.g. model parameter optimization functions) for the low-level phylogenetic likelihood library `libpll` (Flouri *et al.*, 2015). Besides likelihood computations, `libpll` and `pll-modules` libraries

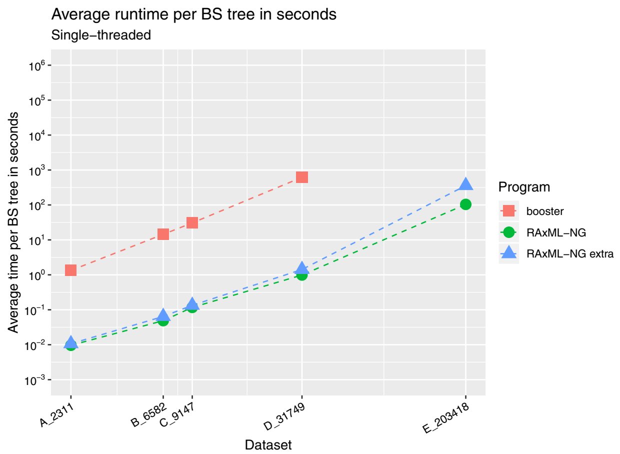

Fig. 1. Average runtime per BS tree in seconds, with and without computing additional statistics in RAxML-NG. Both tools were executed sequentially. Note the logarithmic scale on the *y*-axis. On the E_203418 dataset, booster went out of memory. RAxML-NG is several orders of magnitude faster than booster

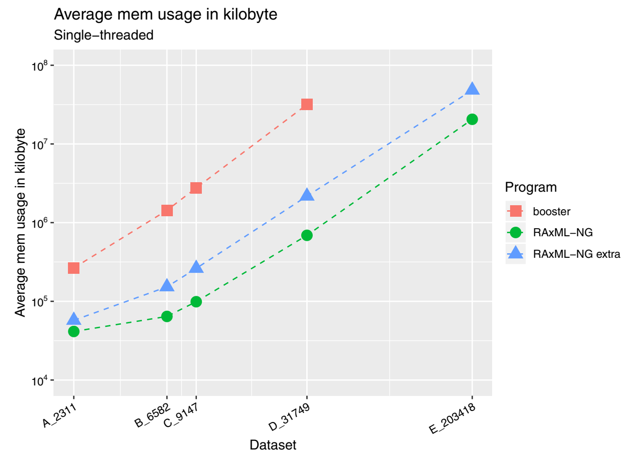

Fig. 2. Average total memory usage in kilobytes, with and without computing additional statistics in RAxML-NG. Both tools were executed sequentially. Note the logarithmic scale on the *y*-axis. On the E_203418 dataset, booster went out of memory. RAxML-NG requires several orders of magnitude less memory than booster across all tested datasets

provide highly efficient tree operations (e.g. NEWICK parsing/writing, tree traversals, manipulating bipartitions). Hence, using `pll-modules` allowed us to leverage these routines for our TBE implementation, and to facilitate integration into third-party programs as well as into our `RAxML-NG` (Kozlov *et al.*, 2019) software.

Our implementation has been integrated into `RAxML-NG v0.8.1` and later versions. We describe our implementation in detail in Supplementary Material.

The `booster` tool calculates additional TBE-related statistics that might help the user to identify potential problems with a dataset, such as the presence of rogue taxa. However, computing these additional statistics increases space- and runtime complexity by a factor of n, where n is the number of taxa. We also designed and implemented an improved algorithm for computing these statistics (see Supplementary Material for details).

3 Results

We compared runtime performance and memory consumption between our implementation and `booster`. Our implementation yields exactly the same scores as `booster` on all datasets we used for this evaluation.

Two other popular phylogenetic inference tools also offer TBE computations: PhyML (Guindon *et al.*, 2010) and IQ-Tree (Nguyen *et al.*, 2015). IQ-Tree internally uses `booster` for this task, and PhyML cannot compute TBE support for user-specified tree sets. Therefore, we excluded IQ-Tree and PhyML from our evaluation.

Note that Truszkowski *et al.* (2019) are simultaneously and independently working on an improved algorithm for TBE computations with a lower theoretical run time complexity. The respective prototype implementation is 237 times faster (personal communication) than the original `booster` implementation on the dataset C. On this dataset, our implementation is 258 times faster and requires 21 times less memory than `booster`.

We measured runtimes and memory consumption on a machine equipped with two Xeon Gold 6148 (Skylake-SP) CPUs and 768GB RAM. Details on the empirical datasets we used for evaluation can be found in Supplementary Material.

Our experimental results show that RAxML-NG is two orders of magnitude faster than `booster` on all datasets, while RAxML-NG uses considerably less memory than `booster` (Figs 1 and 2).

As can be seen in Supplementary Material, RAxML-NG also outperforms `booster` when using multiple threads.

4 Conclusions

We developed and made available a substantially faster and more memory-efficient open source transfer BS implementation. It allows to calculate TBE support metrics on extremely taxon-rich phylogenies, without constituting a computational limitation. For example, using a single thread on dataset D with 31 749 taxa and 100 BS trees, our implementation can compute TBE support values in under 2 min, while `booster` requires 916 min. While we can now rapidly compute the TBE, users should bear in mind that inferring the actual BS trees typically represents the main computational burden of a phylogenetic analysis.

Acknowledgements

We thank Frédéric Lemoine for discussions and feedback on this manuscript.

Funding

Part of this work was funded by the Klaus Tschira foundation.

Conflict of Interest: none declared.

References

Felsenstein,J. (1985) Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, **39**, 783–791.

Flouri,T. *et al.* (2015) The phylogenetic likelihood library. *Syst. Biol.*, **64**, 356–362.

Guindon,S. *et al.* (2010) New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of phyml 3.0. *Syst. Biol.*, **59**, 307–321.

Kozlov,A.M. *et al.* (2019) RAxML-NG: a fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. *Bioinformatics*, **35**, 4453.

Lemoine,F. *et al.* (2018) Renewing Felsenstein's phylogenetic bootstrap in the era of big data. *Nature*, **556**, 452.

Nguyen,L.-T. *et al.* (2015) Iq-tree: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. *Mol. Biol. Evol.*, **32**, 268–274.

Truszkowski,J. *et al.* (2019) Rapidly computing the phylogenetic transfer index. In: Huber, K.T. and Gusfield, D. (eds) *19th International Workshop on Algorithms in Bioinformatics (WABI 2019)*. Volume **143** of *Leibniz International Proceedings in Informatics (LIPIcs)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 20:1–20:12.